

Software AG Command Central Help

Innovation Release

Version 10.2

April 2018

This document applies to Software AG Command Central Version 10.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2013-2018 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://softwareag.com/licenses>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <http://softwareag.com/licenses> and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

Table of Contents

About this Guide.....	13
Document Conventions.....	13
Online Information.....	14
Understanding Software AG Command Central.....	15
What is Software AG Command Central?.....	16
Docker Support.....	16
Online Help.....	17
Installing, Configuring, and Upgrading Command Central.....	19
Automate Command Central Installation, Configuration, and Upgrade.....	20
Install Command Central.....	20
Log On and Start or Stop Command Central.....	23
Change or Reset the Command Central Administrator Password.....	23
Secure the Connection Between Command Central and Platform Managers.....	24
Control User Access to Command Central.....	24
Set Timeouts and Thread Pool Size for Command Central.....	25
Configure Shared Secrets to Encrypt Instance and Component Passwords.....	25
Configure Command Central to Use JAAS for Advanced Authentication.....	26
Upgrade Command Central.....	26
Managing Existing Standalone Product Installations.....	29
Understanding Standalone Product Installations.....	30
Connect an Existing Product Installation to Command Central.....	31
Organize Installations into Environments.....	32
Add an Environment to a Landscape.....	32
Compare Products, Fixes, and Instances.....	32

Change the Status of an Instance or Component.....	33
Tag Instances and Components with Attributes.....	33
Add Licenses and Point to Licenses for Instances or Components.....	33
View KPIs and Alerts for an Instance or Component.....	34
Create, Configure, and Update Product Instances.....	34
Install Fixes.....	34
Install Support Patches.....	35
Before Installing or Upgrading Products.....	37
Understanding Product and Fix Repositories.....	38
Understanding Asset Repositories.....	39
Before Creating Repositories.....	40
Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes.....	40
Create Asset Repositories.....	41
Before Installing Products.....	41
Before Installing Fixes and Support Patches.....	43
Before Creating Database Components.....	44
Creating or Upgrading Standalone Product Installations.....	45
Understanding Standalone Product Installations.....	46
Install and Configure Platform Manager.....	46
About Installing Platform Manager.....	46
Install Platform Manager Using the Command Central Bootstrapper.....	47
Install Platform Manager on Remote Machines.....	48
Install Platform Manager on a Command Central Host.....	48
Configure Platform Manager.....	49
Install Products.....	49
Create Database Components and Connect to Products.....	50
Install Assets.....	50
Upgrade Products.....	50

Creating Software Stacks.....	53
Understanding Software Stacks.....	54
Create a Software Stack.....	55
Install Assets.....	56
Creating and Viewing License Metering Reports.....	57
Understanding License Metering.....	58
Current State of a Landscape.....	58
Continuous License Metering of a Landscape.....	58
Monthly Metering of a Landscape.....	59
Create and View a Snapshot License Report.....	60
Create and View an Aggregated Report.....	60
Create and View a Product Usage Report for Billing.....	60
Tuning and Troubleshooting.....	61
Configure the Command Central and Platform Manager JVMs.....	62
Troubleshooting.....	63
Automation and Template-based Provisioning.....	65
Using Composite Templates.....	67
About Automated Provisioning of Environments.....	68
What is a Composite Template?.....	68
Understanding the Composite Template Definition.....	70
Environments.....	70
Layers.....	73
Templates.....	74
Migration.....	80
Provision.....	86
Nodes.....	88
Repositories.....	92
Actions.....	93
Format of COMMON-* Configuration Types in the Template Definition.....	97
Summary of Sample Composite Templates.....	99
Using the is-layer Sample Template.....	103
Before Using the is-layer Sample Template.....	103
Configuring the is-layer Template.....	104
Understanding the is-layer Template Definition.....	104
Applying the is-layer Sample Template.....	110

How Command Central Processes a Composite Template.....	111
Validating a Composite Template.....	114
Creating a New Environment Using a Composite Template.....	115
Updating a Provisioned Environment Using the Same Composite Template.....	116
Monitoring the Status of a Composite Template.....	117
Correcting a Failed Composite Template Apply Operation.....	117
Developing Composite Templates.....	119
Creating a Custom Composite Template.....	120
Migrating to a Different Host with Cloned Database.....	120
Migrating Using Disconnected Migration.....	121
Managing Database Components Using a Composite Template.....	122
Supported Database Component Configurator Parameters.....	125
Database Actions in the Sample Composite Templates.....	126
Using Micro Templates.....	127
What is a Micro Template?.....	128
When to Use Micro Templates and When a Composite Template?.....	128
Designing Micro Templates.....	129
Using the Sample Micro Templates.....	129
Creating a User-Defined Micro Template.....	130
Validating a Micro Template.....	131
Command Central Developer Reference.....	133
Using the Command Line Interface.....	135
Installing CLI as a Remote Client.....	136
Displaying Help for the Command Line Interface.....	137
Executing Command Central Commands.....	137
Executing Platform Manager Commands.....	138
Getting Familiar with Using Commands.....	138
Return Codes from Command Execution.....	140
Creating Shell Scripts that Execute Commands.....	140
Creating Ant Scripts that Execute Commands.....	141
Parameters to Use with the ccsetup Task.....	141
Parameters to Use with the cc Task.....	143
Using a Unix Shell Script to Change the Administrator Password for Command Central.....	155
Managing Database Components Using CLI Commands.....	157
Supported Database Component Configurator Actions.....	158
Create Database Components.....	158
Drop Database Components.....	159
Migrate Database Components.....	160
Recreate Database Components.....	161
Execute a Database Action with Catalog.....	161
Searching for Database Component Configurator Details.....	162

Options for the Commands.....	165
Common Options.....	167
accept.....	167
check-every.....	168
configuration-file.....	170
debug.....	171
error.....	172
expected-values.....	173
force.....	175
format.....	175
input.....	177
input-format.....	179
log.....	180
media-type.....	181
output.....	182
output-format.....	183
Recommendations When Selecting a Content Type.....	184
password.....	186
properties.....	186
quiet.....	187
retry.....	188
server.....	189
ssl-truststore-file.....	190
ssl-trust-all-hosts.....	191
ssl-truststore-password.....	191
sync-job.....	192
username.....	192
wait.....	193
wait-for-cc.....	194
Administration Commands.....	197
About Administration Commands.....	198
sagcc exec administration.....	198
sagcc list administration.....	200
License Inventory Commands.....	203
sagcc add inventory components contracts assignment.....	204
sagcc update inventory components contracts assignment.....	205
sagcc delete inventory components contracts assignment.....	206
sagcc get inventory components contracts assignment.....	206
Configuration Commands.....	209
sagcc get configuration common.....	210
sagcc get configuration compare.....	211
sagcc create configuration data.....	213

sagcc delete configuration data.....	216
sagcc get configuration data.....	218
sagcc update configuration data.....	221
sagcc get configuration instances.....	225
sagcc list configuration instances.....	227
sagcc get configuration types.....	229
sagcc list configuration types.....	230
sagcc exec configuration validation create.....	233
sagcc exec configuration validation delete.....	235
sagcc exec configuration validation update.....	237
Diagnostics Logs Commands.....	241
sagcc get diagnostics logs.....	242
sagcc get diagnostic logs export file.....	245
sagcc list diagnostics logs.....	247
Instance Management Commands.....	249
sagcc create instances.....	250
sagcc delete instances.....	252
sagcc get instances.....	253
sagcc list instances supported products.....	254
sagcc update instances.....	255
Inventory Commands.....	257
sagcc exec inventory assets refresh.....	258
sagcc get inventory assets.....	258
sagcc list inventory assets.....	260
sagcc create inventory components attributes.....	262
sagcc get inventory components.....	264
sagcc get inventory components attributes.....	266
sagcc list inventory components.....	267
Specifying Search Criteria for Inventory Commands.....	271
sagcc list inventory components attributes.....	273
sagcc update inventory components.....	274
sagcc update inventory components attributes.....	276
sagcc delete inventory components attributes.....	277
sagcc get inventory fixes.....	279
sagcc get inventory fixes compare.....	280
sagcc list inventory fixes.....	282
sagcc get inventory products.....	284
sagcc get inventory products compare.....	286
sagcc list inventory products.....	287
Jobmanager Jobs Commands.....	291
sagcc delete jobmanager jobs.....	292
sagcc delete jobmanager landscapejobs.....	293

sagcc list jobmanager jobs.....	294
sagcc list jobmanager landscapejobs.....	295
Landscape Commands.....	297
sagcc add landscape environments nodes.....	298
sagcc create landscape environments.....	299
sagcc delete landscape environments.....	301
sagcc get landscape environments.....	302
sagcc list landscape environments.....	303
sagcc remove landscape environments nodes.....	304
sagcc update landscape environments.....	306
sagcc create landscape nodes.....	308
sagcc delete landscape nodes.....	310
sagcc exec landscape nodes generateNodeId.....	311
sagcc get landscape nodes.....	313
sagcc list landscape nodes.....	314
sagcc update landscape nodes.....	315
License Keys Commands.....	319
sagcc add license-tools keys.....	320
sagcc get license-tools keys.....	321
sagcc list license-tools keys.....	322
sagcc update license-tools keys.....	325
sagcc delete license-tools keys.....	326
sagcc update configuration license.....	326
License Reports Commands.....	329
sagcc create license-tools reports snapshot.....	331
sagcc delete license-tools reports snapshot.....	331
sagcc delete license-tools reports snapshot reportid.....	332
sagcc list license-tools reports snapshot.....	332
sagcc get license-tools reports snapshot reportid.....	333
sagcc get license-tools reports snapshot output PDF.....	334
sagcc get license-tools reports snapshot output XML.....	335
sagcc create license-tools reports installation.....	336
sagcc delete license-tools reports installation reportid.....	336
sagcc delete license-tools reports installation.....	337
sagcc get license-tools reports installation output PDF.....	337
sagcc get license-tools reports installation output XML.....	338
sagcc get license-tools reports installation disclaimer.....	339
sagcc list license-tools reports installation.....	340
sagcc list license-tools reports.....	340
sagcc list license-tools metering products.....	341
sagcc add license-tools manifests.....	342
sagcc delete license-tools manifests.....	342
sagcc get license-tools manifests output xml.....	343

sagcc list license-tools manifests.....	344
sagcc list license-tools manifests content.....	344
sagcc list license-tools reports aggregated.....	346
sagcc get license-tools reports aggregated reportid.....	347
sagcc add license-tools reports aggregated.....	348
sagcc delete license-tool reports aggregated reportid.....	349
sagcc delete license-tools reports aggregated.....	350
sagcc exec license-tools metering.....	350
sagcc get license-tools metering state.....	351
sagcc get license-tools metering contracts licensekey.....	352
sagcc get license-tools reports aggregated usage reportid.....	353
sagcc list license-tools manifests contracts.....	354
Lifecycle Commands.....	355
sagcc exec lifecycle.....	356
Specifying Search Criteria for Lifecycle Commands.....	359
Monitoring Commands.....	361
sagcc get monitoring.....	362
sagcc get monitoring state.....	364
sagcc list monitoring alerts.....	367
Provisioning Assets Commands.....	369
sagcc exec provisioning assets install.....	370
sagcc exec provisioning assets uninstall.....	372
sagcc list provisioning assets supportedcomponents.....	373
Provisioning Bootstrap Installers Commands.....	375
sagcc create provisioning bootstrap installers.....	376
sagcc list provisioning bootstrap installers.....	378
sagcc exec provisioning bootstrap nodes.....	380
Provisioning Fixes Commands.....	385
sagcc exec provisioning fixes install.....	386
sagcc exec provisioning fixes uninstall.....	387
Provisioning Products Commands.....	389
sagcc exec provisioning products install.....	390
sagcc exec provisioning products uninstall.....	391
Repository Commands.....	393
About Repository Commands.....	395
sagcc add repository assets flatfile.....	395
sagcc add repository assets git.....	398
sagcc delete repository assets.....	401
sagcc list repository assets.....	401
sagcc list repository assets content.....	402

sagcc list repository assets dependencies.....	403
sagcc list repository assets namespaces.....	404
sagcc update repository assets flatfile.....	405
sagcc update repository assets git.....	407
sagcc add repository fixes.....	409
sagcc add repository products.....	412
sagcc delete repository.....	415
sagcc delete repositories.....	416
sagcc exec repository discover.....	417
sagcc exec repository fixes export.....	418
sagcc exec repository refresh.....	419
sagcc list repository.....	420
sagcc list repository discover.....	421
sagcc list repository fixes content.....	422
sagcc list repository fixes dependencies.....	423
sagcc list repository fixes readme.....	424
sagcc list repository products content.....	425
sagcc list repository products dependencies.....	427
sagcc list repository products languages.....	427
sagcc update repository fixes.....	429
sagcc update repository products.....	431
Resources Commands.....	435
sagcc list resources icons.....	436
Security Credentials Commands.....	437
sagcc add security credentials.....	438
sagcc add security credentials sharedsecret.....	441
sagcc delete security credentials.....	443
sagcc get security credentials.....	444
Stacks and Layers Commands.....	447
sagcc create stacks.....	448
sagcc create stacks layers.....	449
sagcc create stacks layers nodes master.....	450
sagcc delete stacks.....	451
sagcc delete stacks layers.....	452
sagcc list stacks.....	453
sagcc list stacks layers.....	454
sagcc list stacks layers fixes.....	454
sagcc list stacks layers fixes compare.....	455
sagcc list stacks layers instances.....	456
sagcc list stacks layers instances compare.....	457
sagcc list stacks layers nodes.....	458
sagcc get stacks layers nodes master.....	459
sagcc list stacks layers products.....	460

sagcc list stacks layers products compare.....	461
Template Commands.....	463
sagcc delete templates composite.....	464
sagcc exec templates composite apply.....	464
sagcc exec templates composite import.....	467
sagcc exec templates composite validate.....	468
sagcc get templates composite export.....	469
sagcc list templates composite.....	470
Configuring Command Central and Platform Manager.....	471
Configuration Types for Command Central and Platform Manager OSGI.....	472
Configuration Types for Command Central and Platform Manager OSGI ENGINE.....	475
Run-time Monitoring States for OSGI-CCE and OSGI-SPM.....	479
Run-time Monitoring States for OSGI-SPM-ENGINE.....	480
Server System Properties.....	481
Authentication.....	481
Migration.....	481
Provisioning.....	482
Prerequisites to Configuring a Port for SSL.....	484
Configuring SSL Using Configuration Properties Files.....	485
Creating a Custom Command Central Properties File.....	485
Considerations When Using Configuration Properties.....	486
Using the Command Central and Platform Manager Logs.....	489
Command Central Logs.....	490
Platform Manager Logs.....	491
Using the Correlation ID.....	493
Logging Levels.....	494
Changing the Log Configuration Settings.....	494
Changing the Wrapper Logs Configuration.....	495
Deleting Logs.....	495
Size-based Log Rotation (Default).....	495
Delete Command Central Logs.....	495
Delete Platform Manager Logs.....	496
Time-based Log Rotation.....	497
Introduction to Command Central REST API.....	499
About Command Central REST API.....	500
Securing Command Central REST API.....	500
Session Management.....	500
Command Central REST API Resources.....	500
Supported Media Types.....	501
HTTP Response Codes.....	502
Summary of REST Services.....	502

About this Guide

This guide explains how to use Software AG Command Central to manage your Software AG products remotely from one location.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention <i>folder.subfolder:service</i> .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
<i>Italic</i>	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 (...).

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <http://documentation.softwareag.com>. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at <https://empower.softwareag.com>.

To submit feature/enhancement requests, get information about product availability, and download products, go to [Products](#).

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the [Knowledge Center](#).

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

I Understanding Software AG Command Central

■ What is Software AG Command Central?	16
■ Docker Support	16
■ Online Help	17

What is Software AG Command Central?

Software AG Command Central is a tool that enables you to install, patch, configure, manage, and upgrade Software AG products; create database components; and connect products to database components, remotely from one location. Command Central offers a browser-based user interface, but you can also automate tasks by using commands to remotely execute actions from a terminal or custom script (for example, CI servers such as Jenkins or generic configuration management tools such as Puppet or Chef).

This release of Command Central can manage Software AG products that are release 9.0 or later. Software AG recommends always using the newest Command Central with your products so you can take advantage of the most up-to-date features as well as stability, usability, and security enhancements. If you have an older Command Central, you can upgrade to the newest release even if you are not upgrading your products. See the *Software AG Command Central Feature Support Matrix* for the list of products you can manage using Command Central and the features that are supported for each release of those products.

With Command Central, you can work with existing standalone product installations or create new ones. If you are following DevOps practices, you can create software stacks of product runtimes. For each case, you create repositories from which to install products (9.8 and later), install fixes (9.7 and later), and deploy assets (10.1 and later).

You install one Command Central on a machine that has Internet access. This Command Central connects to the Software AG Software Download Center (SDC) and the Empower Product Support website so you can download the products and fixes you have licensed.

You install one or more Command Centrals to manage your development and test environments, and one or more Command Centrals to manage production environments. These Command Centrals do not need Internet access but must have access to the machines that host the products you want to manage.

Platform Manager is the agent for Command Central. Platform Manager is installed with every Command Central and in every product installation. When you submit requests for actions against a product installation, Command Central directs the requests to the Platform Manager in that installation, and the Platform Manager executes the action. Platform Manager is always the same release as the products.

Docker Support

Software AG ships an official Docker image of Command Central on Docker Store <https://store.docker.com/images/softwareag-commandcentral>

Using the image, you can get Command Central up and running quickly and easily. You can also use the Command Central Docker image for template development. For

documentation and examples, go to <https://github.com/SoftwareAG/sagdevops-hello-docker>

Command Central also ships Docker Builder, which you can use to build custom Docker images for some Software AG Digital Business Platform products. For an example, go to <https://github.com/SoftwareAG/sagdevops-cc-docker-builder>

Use of Command Central in this way assumes experience with Docker.

Online Help

The Command Central graphical user interface provides instructions and tooltips to help you perform tasks. Instructions for performing tasks using the GUI is provided in this online help only when additional information is necessary.

II Installing, Configuring, and Upgrading Command Central

■ Automate Command Central Installation, Configuration, and Upgrade	20
■ Install Command Central	20
■ Log On and Start or Stop Command Central	23
■ Change or Reset the Command Central Administrator Password	23
■ Secure the Connection Between Command Central and Platform Managers	24
■ Control User Access to Command Central	24
■ Set Timeouts and Thread Pool Size for Command Central	25
■ Configure Shared Secrets to Encrypt Instance and Component Passwords	25
■ Configure Command Central to Use JAAS for Advanced Authentication	26
■ Upgrade Command Central	26

Automate Command Central Installation, Configuration, and Upgrade

The topics in this help explain how to install, configure, and upgrade Command Central using a step-by-step procedure. You can automate these tasks instead. For details, go to www.github.com/SoftwareAG/sagdevops-cc-server.

Install Command Central

You install Command Central using the Command Central bootstrapper. The bootstrapper also automatically installs Platform Manager and the Command Central command line interface (CLI), as well as all fixes that were available for Command Central and the other tools when the bootstrapper was created.

Each time a new fix for Command Central or the other tools is created, the bootstrapper is refreshed to include the new fix. To install new fixes, get the latest bootstrapper for the same Command Central release and run it against the existing Command Central installation. Whenever you install a new Command Central, always use the latest bootstrapper, to get the most recent fixes.

Machines on which you install Command Central need at least 5GB of disk space. Command Central then needs 2GB of RAM and 4 CPU cores to run. If you plan to install products, install fixes, or upgrade products from Command Central, you will need an additional 50GB of disk space. Command Central host machines must have access to the host machines of all products you want to manage. Restrict user access to only those people who need to use Command Central.

Important: Do not install any other Software AG products in the Command Central directory. One reason is to enable you to restrict access. A second reason is that you should upgrade Command Central when new releases come out, even if you are not yet upgrading your products, and keeping Command Central separated from the products makes upgrading it easier.

For supported operating systems and browsers, see *System Requirements for Software AG Products*.

You can watch a demo relating to this task in the Command Central area of the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>.

Note: To upgrade Command Central and the other tools to a new release, see the upgrade topic in this help.

1. Read the Software AG product license at http://documentation.softwareag.com/legal/general_license.txt. You will need to accept the license when you run the bootstrapper.

2. If you are going to install on a Windows system:

- Make sure the machine has the latest Windows updates. On Windows 8.1 and Windows Server 2012 R2, installation will fail if the Microsoft update KB2919355 from April 2014 is not present.
- Create an installation user account and give the account Windows system administrator privileges. This account will own all files you install.

If you are going to install on a UNIX system, create a non-root installation user account and give the account full read and write permissions to the target installation directory. If you want to register daemons for Command Central and Platform Manager, ask your system administrator for the sudo password. The account will own all files you install.

3. Log on to the target machine under the account you just created.
4. Open your installation email from Software AG and follow the instructions to download the Command Central bootstrapper for the appropriate operating system. If you are installing on a Windows system, extract the ZIP archive to the directory into which you are going to install Command Central.

Note: If you download the UNIX bootstrapper .sh script and then transfer it to another machine, set the transfer tool you are using to binary mode.

5. The installation process produces many messages. To capture all messages, either increase your buffer or send the output to a file by specifying `--params > bootstrapper.out` on the command below.
6. Open a command window as Administrator for Windows or a shell window for UNIX and run the .bat file or .sh script with the arguments in the table below. Arguments that have defaults are optional.

Argument	Value	Default
<code>-H <i>host_name</i></code>	DNS name or IP address for the machine on which to install Command Central.	Machine on which you are running the bootstrapper.
<code>-d <i>path</i></code>	Full path to the installation directory in which to install Command Central.	Directory from which you are running the bootstrapper.
<code>-c <i>port_number</i></code>	HTTP port to use for Command Central.	8090

Argument	Value	Default
<code>-C port_number</code>	HTTPS port to use for Command Central.	8091
<code>-s port_number</code>	HTTP port for Command Central to use to communicate with the local Platform Manager.	8092
<code>-S port_number</code>	HTTPS port for Command Central to use to communicate with the local Platform Manager.	8093
<code>-p password</code>	Password to use for the Command Central Administrator user account. If you are installing in a production environment, Software AG strongly recommends you replace the default with a strong, custom password.	manage
<code>-P sudo_password</code>	If you are installing on a UNIX system and want to register daemons for Command Central and Platform Manager, the sudo password.	
<code>--accept-license</code>	Indicates that you accept the Software AG product license.	

- After installation is complete, Command Central is running and ready for use. The bootstrapper lists the Command Central URL and authentication credentials in the command or shell window. Note this information so you can log on.

Note: If you did not register daemons when you installed Command Central but want to do so at a later time, you can run the bootstrapper again with the argument `-P sudo_password`.

- If the Command Central managing an environment, secure access to the Command Central. This is especially important if you are using the Command Central to manage production environments.

Examples

- To install on a Windows system in the `c:\sagcc` directory and set the password for the Command Central Administrator user account to `$SuperCCAdmin`:

```
unzip cc-def-10.2-fix1-w64.zip
```

```
cc-def-10.2-release-w64.bat -d c:\sagcc -P $SuperCCAdmin --accept-license
```

- To install on a UNIX system in the /opt/sagcc directory on the UNIX system cchost.com, set the port values, and set the password for the Command Central Administrator user account to \$SuperCCAdmin:

```
chmod +x cc-def-10.2-fix1-lnxamd64.sh
./cc-def-10.2-fix1-lnxamd64.sh -d /opt/sagcc -H cchost.com
-c 9090 -C 9091 -s 9092 -S 9093 -P $SuperCCAdmin --accept-license
```

Log On and Start or Stop Command Central

After installation or upgrade is complete, Command Central is running and ready for use.

Open an Internet browser and specify the URL that the Command Central bootstrapper listed in the command or shell window at the end of the installation process, as follows:

`http://Command_Central_host:Command_Central_port`

On a Windows system, you can start or stop Command Central and Platform Manager using their Windows services.

On a UNIX system, you can start or stop Command Central and Platform Manager using the startup.sh or shutdown.sh script in the *Software AG_directory*/profiles/CCE/bin and *Software AG_directory*/profiles/SPM/bin directories, respectively.

Change or Reset the Command Central Administrator Password

When you installed Command Central, you specified a strong, custom Administrator password. Software AG strongly recommends you change the Command Central Administrator password periodically.

Tip: If you forget the password, you can use the Command Central bootstrapper to reset it. You can also use the bootstrapper to automate changing the password for both the Command Central and the local command line interface (CLI) password at the same time by running the bootstrapper with the arguments `-d same_installation_directory -p new_password`.

1. Go to **Environments > ALL > Instances > CCE > Configuration > Internal Users > Administrator** and edit the password.
2. Verify the new password by logging out and then logging in with the new password.
3. To update the Command Central command line interface configuration file:
 - a. Run a command (for example, `sagcc list landscape nodes`). The command returns ERROR 401.

- b. Open the `$HOME\.sag\cc.properties` file in a text editor and set the `password` property to the new password.
- c. Verify the new credentials by running the command again.

Secure the Connection Between Command Central and Platform Managers

Command Central comes with default certificates on both sides of the connection between Command Central and Platform Manager. Software AG recommends you replace these with your own certificates.

You can watch a demo relating to this task in the Command Central area of the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>. You can also read an article on generating keystores and certificates for Command Central in the same area.

1. For Command Central, go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > General Properties > Outbound SSL Connection Settings** and then click **Edit**.
2. For Platform Manager, go to **Environments > ALL > Instances > SPM > Configuration > Ports > defaultHTTPS** and provide the requested values in the **Security Configuration** area.

Control User Access to Command Central

Command Central uses users, groups, and roles to authenticate users and determine the actions they can perform. Command Central supports read, write, execute, and password read permissions.

You can define users and groups in Command Central's internal user repository, or you can use users and groups from Lightweight Directory Access Protocol (LDAP) or Microsoft Active Directory (AD) acting as an LDAP server, or both. Command Central can work with multiple LDAP or AD user stores.

The permissions you set up for a Command Central apply across the entire landscape managed by that Command Central, which means that a user or group has the same permissions for all environments managed by that Command Central. If you want a user or group to have different permissions for different environments, install a Command Central to manage each environment.

Software AG recommends defining and implementing your authorization model and then not changing it. In production, the only change that should occur is assigning users to groups, which is normally done when LDAP or AD is implemented.

Note: You do not need to define users, groups, and roles for Platform Manager unless you are using third-party monitoring software that communicates directly with Platform Manager. In this case, add an internal user and assign a

role that has canread permissions to that user. Use the instructions below, but for Platform Manager instead of Command Central.

1. To add users, go to **Environments > All > Instances > CCE > Configuration > Internal Users**, click , and provide the requested values.
2. To add groups, go to **Environments > All > Instances > CCE > Command Central Server > Configuration > Internal Groups** and click **Edit**.
3. If you want to connect to LDAP, go to **Environments > All > Instances > CCE > Configuration > LDAP**, click , and provide the requested values.
4. To add roles, and then assign them to groups and users, go to **Environments > All > Instances > CCE > Command Central Server > Configuration > Security Roles** and click **Edit**.

Set Timeouts and Thread Pool Size for Command Central

To set timeouts, go to **Environments > ALL > Instances > CCE > Configuration > Java System Properties** and configure the properties as described.

Configure Shared Secrets to Encrypt Instance and Component Passwords

When you want to view or update instance or component configurations that contain passwords, you can provide a shared secret for Platform Manager to use to create a security context and encrypt the passwords.

You can specify an environment shared secret for any environment, and a global shared secret for an entire Command Central landscape. If an instance or component is in an installation that belongs to a single environment, Command Central uses the shared secret for that environment. If the instance or component is in an installation that does not belong to an environment or that belongs to several environments, Command Central uses the global shared secret.

Command Central is installed with a default global secret, but Software AG strongly recommends that you replace it. If you do not, it will be possible to decrypt sensitive information in templates or configurations using the default Command Central password.

1. To specify an environment shared secret, go to **Environments**, click the environment for which to specify the shared secret, click  , click **Configure Shared Secret**, and type the shared secret to use.
2. To specify a global shared secret, go to **Environments > ALL**, click  , click **Configure Shared Secret**, and type the shared secret to use.

Configure Command Central to Use JAAS for Advanced Authentication

Command Central comes with default login modules that perform user authentication. You can use Java Authorization and Authentication Service (JAAS) to create and deploy additional, custom login modules. You would only need custom login modules for Command Central if you are setting up an advanced security scenario, such as using a third-party security provider. For instructions on creating and deploying custom login modules, see *Software AG Infrastructure Administrator's Guide*.

Important: Do not remove the default login modules that come with Command Central.

To use JAAS, go to **Environments > All > Instances > CCE > Command Central Server > Configuration > JAAS Realms**.

When you connect to an LDAP or AD user store, Command Central automatically updates the JAAS configuration to use it.

Upgrade Command Central

When you upgrade Command Central, these configurations, data, and assets are migrated:

- Binary cache.
- Composite templates.
- Credentials.
- Environment, JAAS, LDAP, port, and proxy configurations.
- HTTPS and SSO certificates.
- License reports.
- Primary and image repositories.
- Users, groups, and roles, and password manager data.

Follow the instructions below to upgrade.

1. If you are going to install the new Command Central on a different machine than the old Command Central, create a ZIP file of the old Command Central. These instructions use the Java Archive tool to create the ZIP file.
 - a. Go to the old Command Central machine.
 - b. In the `JAVA_HOME` and `PATH` system variables, specify the location of the Java Archive tool as `Software AG_directory \jvm \jvm \bin`.

Note: On some systems, the lower-level `jvm` directory name includes additional information, such as `\jvm\jvm160_32`, or `\jvm\jvm170`, or `\jvm\jvm_64`.

- c. Open a command window or shell and go to the Software AG directory that contains the old Command Central.
- d. If you want to reduce the size of the ZIP file, move the log files out of the `old_Software_AG_directory\profiles\CCE\logs` and `SPM\logs` directories.
- e. Enter this command:


```
jar cfM ZIP_file_name common\conf profiles\CCE profiles\SPM install\products
```
- f. Copy the ZIP file to any directory on the new Command Central machine.

Important: If using FTP to copy, use the binary file transfer mode\type. If you use another mode\type, the ZIP file might become corrupted.

2. Use the Command Central bootstrapper to install Command Central and to migrate configuration files and assets from the old installation to the new installation. Follow the instructions in the topic in this help for installing Command Central with these additional guidelines:

- Specify the argument `-m old_Command_Central`, as follows:

If <code>old_Command_Central</code> is on...	Specify...
Same machine as new installation	Software AG directory that contains the old Command Central installation.
Different machine than new installation	ZIP file you created in the previous step.

- You must specify the argument `-p password`. As always, if you are installing in a production environment, Software AG strongly recommends you replace the default with a strong, custom password.
- 9.12 and higher: If you want to use the same ports for the new Command Central that you used for the old one, do not specify the `-c` and `-C` port arguments; the bootstrapper will migrate the port configurations and HTTPS and SSO certificates. If you want to use new ports for the new Command Central, specify the `-c` and `-C` port arguments; the bootstrapper will not migrate any port information.

If the old Command Central used non-default ports to communicate with the local Platform Manager, specify the non-default ports on the `-s` and `-S` port arguments.

Note: If you do not specify the non-default ports in this scenario, the local Platform Manager will use the default ports. However, the migrated Command Central configuration contains the non-default ports, so Platform Manager will appear offline and you will have to specify the non-default ports in the new Command Central.

3. 9.7 upgrade: If you set a custom password for Platform Manager, set the custom password for the new Platform Manager. To do so, in the new Command Central, on the Instances tab, click the new Platform Manager (SPM), then click  next to **Authentication** on the Overview tab.

III Managing Existing Standalone Product Installations

- Understanding Standalone Product Installations 30
- Connect an Existing Product Installation to Command Central 31
- Organize Installations into Environments 32
- Add an Environment to a Landscape 32
- Compare Products, Fixes, and Instances 32
- Change the Status of an Instance or Component 33
- Tag Instances and Components with Attributes 33
- Add Licenses and Point to Licenses for Instances or Components 33
- View KPIs and Alerts for an Instance or Component 34
- Create, Configure, and Update Product Instances 34
- Install Fixes 34
- Install Support Patches 35

Understanding Standalone Product Installations

If you have existing product installations that were created using the Software AG Installer, you can quickly connect Command Central to those product installations and organize them into development, test, and production environments. You can then easily monitor and maintain the product instances in your installations. You can:

- Compare instance configurations across installations and environments.
- Change the status of an instance (start, stop, and so on).
- Install fixes and support patches on instances.
- View key performance indicators (KPIs) and alerts. Three KPIs are provided for an installation: system CPU, disk space, and system memory usage. Each KPI shows a marginal threshold, which indicates that performance or stability might soon be affected, and a critical threshold, which indicates that performance or stability are probably affected. Alerts are raised when the value of a KPI changes from normal to marginal or critical, or from marginal to critical. Alerts are cleared when KPI values return to normal.

Up to three KPIs are provided for some, but not all, instances and components when they are online. Command Central retrieves KPIs regularly from the instance or component by polling. Alerts are raised when the status of an instance or component changes from online to stopped, unresponsive, failed, or unknown, or when a KPI value changes from normal to marginal or critical or from marginal to critical. Alerts are cleared when status or KPI values return to normal.

The set of environments that are managed by a particular Command Central is called a *landscape*.

You can perform these tasks from the Command Central GUI or by running Command Central commands.

You can automate maintenance tasks for product environments by developing *composite templates* that define an environment using domain specific language (CC DSL) and applying the templates using commands. Templates can update existing environments with new fixes, instances, configurations, and files.

Note: The term "instance" as used in this section includes both runtime instances and runtime instance components. The latter are independent modules that run within a runtime instance but have their own configurable elements. For example, Task Engine is a runtime instance component on My webMethods Server.

Connect an Existing Product Installation to Command Central

1. Go to the *Software AG_directory \profiles* directory.
 - If you do not see an SPM directory, install Platform Manager in the product installation using the Software AG Installer. You will find Platform Manager at the top level of the installer product selection tree, or under Infrastructure, depending on your release. Note the Platform Manager port numbers. The Software AG Installer starts Platform Manager automatically after installation.
 - If you do see an SPM directory, go to the configuration *\com.softwareag.platform.config.propsloader* directory and note the Platform Manager port numbers on the files named *com.softwareag.catalina.connector.http.pid-port.properties* and *com.softwareag.catalina.connector.https.pid-port.properties*. If Platform Manager is not running, go to the *profiles/SPM/bin* directory and start Platform Manager by running the *startup.{bat|sh}* script.
2. Go to **Environments > ALL > Installations**, click  , click **Add Installation**, and provide the requested values. Command Central automatically adds the installation to **Environments > ALL**.
3. When managing product instances and components of release 9.8 and later (or My webMethods Server 9.9 and later), Command Central uses trusted authentication, which is based on SAML 2.0. When managing instances and components of earlier releases, however, Command Central uses basic authentication. If the instance or component is using non-default credentials for basic authentication, Command Central must supply those non-default credentials to connect. In such cases, do the following:
 - a. On the Instances tab, click the instance or component for which to supply non-default credentials.
 - b. On the Overview tab, if the status shows that the instance or component is not running, start it. Click  every few seconds to refresh the status.

If the status shows Starting and then Online, Command Central is using the correct credentials to connect. If the status shows another status such as Unresponsive, click the Logs tab and open the logs; they will contain errors that say access was denied. In this case continue to the next step.
 - c. Click  next to **Authentication**, then click **Fixed User** and supply the non-default user name and password.
 - d. Click  . After a few seconds, the status should show Online and KPIs should display.

- e. Restart the instance or component and click each status in turn to make sure they report correctly. Check the logs again to make sure there are no access denied errors.

Organize Installations into Environments

When you connect Command Central to an existing installation, Command Central automatically adds the installation to **Environments > ALL**. You can then add the installation to one or more environments you define.

If you later remove an installation from every user-defined environment, that installation is still listed in the **ALL** environment unless you explicitly remove it from the ALL environment.

1. Go to **Environments > ALL > Installations**.
2. Drag and drop installations onto the target environment in the Environments pane. To select multiple installations, hold down the Shift or Ctrl key.

Add an Environment to a Landscape

1. Go to **Environments > ALL**.
2. On the Environments pane, click  and provide the requested values.
3. If you later need to edit the values, click the environment, click  on the Environments pane, and click **Edit Environment**.

Compare Products, Fixes, and Instances

1. To compare products or fixes, go to **Environments > ALL > Installations**.
 - a. Click the rows for up to five installations. To select multiple installations, hold down the Shift or Ctrl key.
 - b. Click  and select **Compare Products** or **Compare Fixes**.
2. To compare instances, go to **Environments > ALL > Instances**.
 - a. Select the rows for up to five instances. To select multiple instances, hold down the Shift or Ctrl key.
 - b. Click , click **Compare Configurations**, and click the type of configuration to compare in the drop-down list.

Change the Status of an Instance or Component

1. Go to **Environments > All > Instances**.
2. Open the tree for the instance, click the Status icon for the instance or component, and click the desired action. Different actions are available for different instances. Most actions self-explanatory. Pause and resume work differently for different product instances; see *Administering Software AG Products Using Command Central* for details.
3. To view the log for a product instance, click the Logs tab. If you want to download logs (for example, because you want to compare them, or use tools to process them), do the following:

Download	Action
One log	Click  for that log.
Multiple logs	Hold down the Shift or Ctrl key and select the log rows. Click  and then click Download selected logs .
All logs	Click Download selected logs without selecting any log.

Tag Instances and Components with Attributes

You can tag instances and components with attributes so you can run inventory or lifecycle commands against sets of instances and components that have those attributes. For example, you could tag instances with an attribute named `environment` and set the attribute to `dev`, `test`, or `prod`.

1. Go to **Environments > ALL > Instances** and click the instance or component for which to add attributes.
2. Click the Overview tab and add the attributes.

Add Licenses and Point to Licenses for Instances or Components

Software AG requires license keys for some products. Software AG provides license key keys for these products when you first license the product, when you need to replace license keys that are about to expire, or when you need to change your license so you can access different product features.

1. To add product license keys to Command Central, go to **Licensing > Keys**, click , choose the appropriate option, and provide the requested values.
2. To point to the license key for an instance or component:
 - a. Go to **Environments > All > Instances** and click the instance or component that requires the license key.
 - b. Click **Configuration**, click **Licenses**, click the license key, click **Edit**, and click the license key to use.

View KPIs and Alerts for an Instance or Component

1. Go to **Environments > ALL**.
2. To check alerts and KPIs for an installation, click the Installations tab and then click the installation.

Note: On Mac OS X and some Linux systems, system memory often shows close to 100% utilization even under normal conditions due to the way Linux manages the memory.

3. To check alerts and KPIs for an instance or component, click the Instances tab, and then click the instance or component.

Note: Not all instances and components provide KPIs.

4. The default polling interval is 30 seconds. To change this interval, go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > General Properties > Monitoring Settings** and click **Edit**.

Note: More frequent polling will adversely affect performance.

Create, Configure, and Update Product Instances

For instructions on creating, configuring, and updating product instances, see *Administering Software AG Products Using Command Central*.

Install Fixes

You can install multiple fixes in an installation at the same time.

You can watch a demo relating to this task in the Command Central area of the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>.

Depending on the number of fixes you select for installation, the fix installation job could take some time to complete.

Note: Fixes released after the Standard Maintenance period are only available via Command Central to customers who have extended maintenance contracts with Software AG. Therefore you might see more fixes when you go to the Empower Fix Explorer page than are available to you through Command Central.

1. Create a fix repository that contains the fixes to install. For instructions, see the fix repository concept information and instructions in this help.
2. Go to **Environments > ALL > Installations** and click the target installation.
3. On the Fixes tab, click , click **Install fixes**, click the fix repository to use, and select the fixes to install.
4. If you later need to uninstall the fix, select the fix and click . When you uninstall a fix, Command Central rolls the product back to the previously installed fix.

Install Support Patches

You can install one support patch at a time. You will need the support patch key from Software AG Global Support.

1. Create a fix repository that contains the support patches to install. For instructions, see the fix repository concept information and instructions in this help.
2. Go to **Environments > ALL > Installations** and click the name of the target installation.
3. On the Fixes tab, click , click **Install support patch**, click the fix repository to use, and click **Next**.
4. Enter the support patch key.
5. If you later need to uninstall the support patch, select the support patch and click .

IV Before Installing or Upgrading Products

■ Understanding Product and Fix Repositories	38
■ Understanding Asset Repositories	39
■ Before Creating Repositories	40
■ Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes	40
■ Create Asset Repositories	41
■ Before Installing Products	41
■ Before Installing Fixes and Support Patches	43
■ Before Creating Database Components	44

Understanding Product and Fix Repositories

The Software AG Software Download Center (SDC) hosts repositories that contain all Software AG products. The Empower Product Support website hosts repositories that contain all Software AG fixes. On the Command Central that has Internet access, you define a connection from the local Platform Manager to these repositories. This connection enables you to install products and fixes on target installation machines.

The connection also enables you to create *mirror* repositories from which to install products and fixes on target installation machines in distant locations or in environments that do not have Internet access. On the Command Central that has Internet access, you first create mirror repositories that serve as the primary source of products and fixes you have licensed, and store those primary mirror repositories locally. From Command Centrals without Internet access, you then connect to the Command Central that has Internet access, create secondary mirror repositories from the primary mirror repositories, and store the secondary mirror repositories as follows:

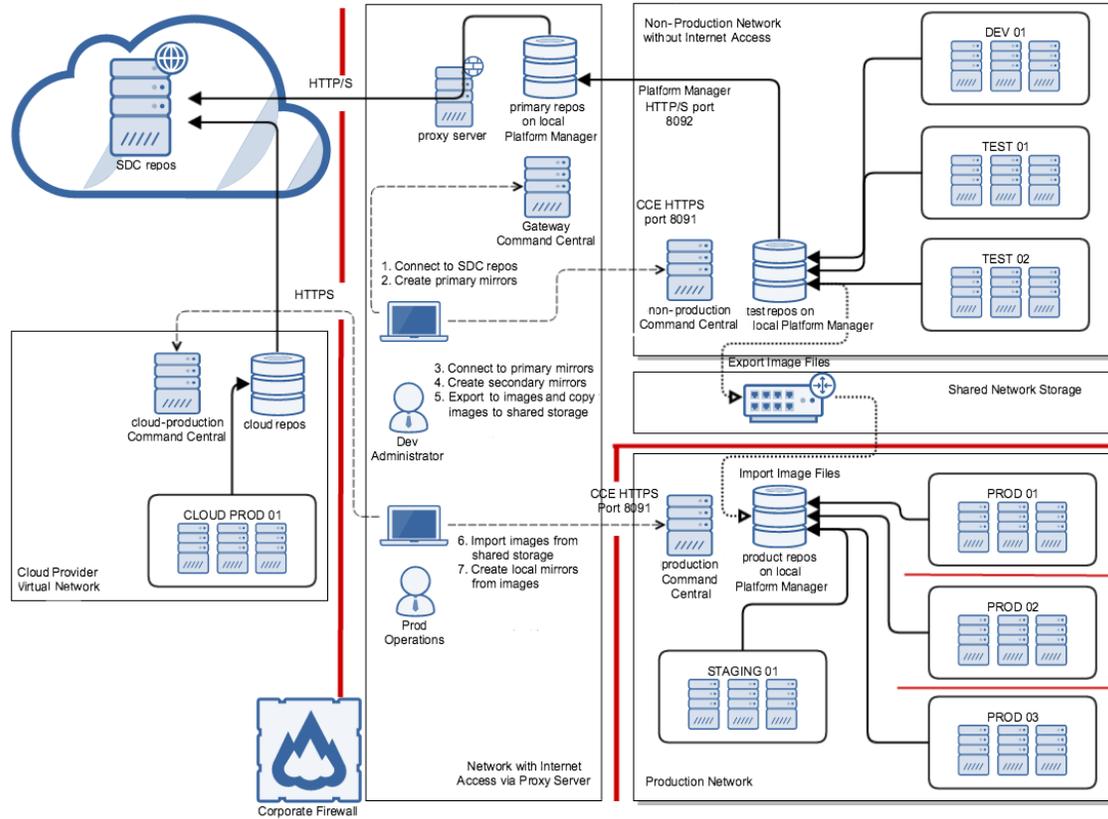
- If the Command Central machine without Internet access is physically near the target installation machines, you store the repositories on that Command Central machine.
- If the target installation machines are in a distant location, you can improve performance by storing the repositories on a machine that is physically near the target installation machines.

From any Command Central, you can connect to mirror repositories that are managed by another Command Central to install products and fixes. For example, you might want to connect from a Command Central in a production environment to a Command Central in a test environment to re-use product and fix binaries that have been tested and are ready for production. Mirror repositories are always hosted on Platform Manager.

You can refresh an existing mirror repository whenever new products or fixes become available in the source repositories. You can also specify different source repositories from which to populate a mirror repository.

If a Command Central without Internet access cannot connect to the Command Central that has internet access, you use *image* repositories to install products and fixes. An image repository is an image file you create from a secondary mirror repository using a Command Central command, from a product installation image you create using Software AG Installer, or from a fix installation image you create using Software AG Update Manager, and then upload to Command Central. For example, if your production environment has no Internet access, you would create an image repository in your test environment, upload it to the Command Central in the production environment, create a mirror repository using the image repository as source, and then install from the mirror repository. You can then delete the image repository.

Note: Software AG strongly recommends creating mirror repositories from image repositories and installing from the mirror repositories. The use of mirror repositories greatly reduces transfer times and improves performance.



Understanding Asset Repositories

Asset repositories contain *composites*. Each composite contains assets and metadata for one type of server (for example, webMethods Integration Server) that you deploy onto a runtime instance of that server type.

You use your continuous integration server and your version control system with webMethods Asset Build Environment to build assets into composites and to store those composites in an asset repository. Asset Build Environment produces a flat file asset repository, which consists of a directory of asset-related files such as asset composite definition language (ACDL), ZIP, and jar files. You can point to the directory from Command Central or ZIP the directory and upload it to Command Central. You can also push the flat file repository files to a Git repository on a machine accessible to Command Central and then connect Command Central to the Git repository. For information on pushing file to Git repositories, see the vendor documentation.

Before Creating Repositories

To connect to product, fix, or asset repositories from Command Central, you must supply the appropriate credentials. Create aliases for credentials as explained below.

1. Go to Empower, log on with the user name and password from the installation email sent to you by Software AG, and read the license agreement. If you accept, continue to the next step.
2. Go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials**, click , and follow the steps below.

To create a credentials alias for connecting to...	Follow these steps...
Software AG product and fix repositories	Click Username and password and create an alias for an Empower username and password that can access the Software AG Software Download Center.
Mirror repositories that were created on other Command Centrals and that are hosted on remote Platform Managers	Click Username and password and create an alias for the username and password that can access the remote Platform Manager.
Git asset repositories	Click Username and password and create an alias for a Git username and password (HTTP/S repository URL), or click SSH private key and create an alias for SSH credentials for accessing Git (SSH repository URL).

Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes

You can watch a demo relating to this task in the Command Central area of the Software AG TECHcommunity website at <http://techcommunity.softwareag.com>.

1. If you are going to create a secondary mirror repository and store it on a machine that is physically near the target installation machines, install a Platform Manager on the machine that will host the repository.

2. Go to **Repositories**, click the appropriate tab, click , click **Create mirror repository**, and provide the requested values.
3. If you later need to refresh the repository with the latest products or fixes, click  and then click **Refresh Repository**.
4. If you later need to specify different source repositories, click , click **Edit Repository**, and specify the new source repositories.

Create Asset Repositories

Use your continuous integration server and your version control system with Asset Build Environment to build assets into composites and to store those composites in an asset repository. Asset Build Environment produces a flat file asset repository, which consists of a directory of asset-related files such as asset composite definition language (ACDL), ZIP, and jar files. You can point to the directory from Command Central or ZIP the directory and upload it to Command Central. You can also push the flat file repository files to a Git repository on a machine accessible to Command Central and then connect Command Central to the Git repository. For information on pushing files to Git repositories, see the vendor documentation.

Before Installing Products

- Work with your administrators, Software AG Global Consulting Services, and best practices documentation to plan a production environment.
- For information on supported operating systems, see the system requirements for your products.

Important: Command Central does not distinguish among versions (Windows) or flavors (UNIX) of an operating system. Make sure you install products only on the supported versions or flavors listed in the products' system requirements.

- For information on product hardware requirements and instructions on preparing the target machines for product installation, see the installation guide for your products.
- Open your installation email from Software AG. Copy the following attachments to a machine that can access the Command Central from which you will install the products:
 - Copy the product license files. Add the product licenses to Command Central by going to **Licensing > Keys**, clicking , choosing the appropriate option, and providing the requested values.

- If you must or want to do license metering for the installation, copy the license manifest files. Go to **Licensing > Manifests**, clicking , and provide the requested values.
- If you are going to install on Windows systems:
 - Create an installation user account on each target machine and give the accounts Windows system administrator privileges. This account will own all files you install.
 - If you are going to install remotely from Command Central on Windows 2008 or higher systems, enable remote access on each target machine using Windows Remote. You must also have Powershell 5.0 or higher and DotNet 4.5 or higher.
 1. Open a Powershell window as administrator and run this command:

```
PS> Enable-PSRemoting -SkipNetworkProfileCheck
```
 2. Increase the Powershell script memory by running this command:

```
PS> Set-Item WSMan:\localhost\Shell\MaxMemoryPerShellMB 2048
```
 3. Adjust your firewall rules to allow access to the HTTP/S ports on which Command Central is listening.
 4. Make sure the machines have the latest Windows updates. On Windows 8.1 and Windows Server 2012 R2, installation will fail if the Microsoft update KB2919355 from April 2014 is not present.

Note: You can only install Software AG products on a local hard drive on Windows machines. You cannot install the products on a network-mounted drive.

If you are going to install on a UNIX system:

- Create a non-root installation user account on each target machine to own the directory that will contain the product installation, but that otherwise has minimal rights on the target machine. The account must have write and execute access to the target installation directory. The account will own all files you install.
- If you are going to install remotely from Command Central, give the user accounts SSH privileges.
- If you are going to register a daemon for the Platform Manager on each machine, ask your system administrator for the sudo password.
- Keep the following in mind when installing:
 - To avoid problems with shared system resources, run only one installation job at a time on the machines. Make sure the entire product installation is complete before starting any other installation jobs on those machines.
 - If your temporary directory contains thousands of files, the installation startup process might take one minute or longer. You might see messages about

- Initializing system resources during this time. If you want to speed up this process, delete the files in your temporary folder.
- Only install products from a single release in an installation directory. Do not mix products from different releases in the same installation directory, or you will experience problems and be unable to access functionality.
 - If you want to use a symlink for the installation directory, you must use the symlink every time you install into the directory. For example, you cannot install some products using a directory name and then later create a symlink and use it to install more products into the same directory, or vice versa.
 - The installer installs a JDK for the products. Do not apply maintenance updates from the vendor of the JDK. If an update is required, Software AG will provide the update in the form of a fix.
 - Do not modify or remove files that are installed or created by Command Central when installing products unless specifically instructed to do so by Software AG. Do not modify or remove files from the *Software AG_directory/install* directory.

Before Installing Fixes and Support Patches

- Do not modify or remove files that are installed or created by Command Central when installing fixes or support patches unless specifically instructed to do so by Software AG. Do not modify or remove files from the *Software AG_directory/UpdateManager* directory.
- Back up the *Software AG_directory/Update Manager* installation directory regularly, in synchronization with your regular backups of your Software AG products. These backups ensure that Software AG can restore the products as well as the Update Manager metadata to a known point of time.

Before installing a fix or support patch on a product, back up both the *Software AG_directory/Update Manager* installation directory and the product directory. Before installing a fix in a production environment, verify the fix in a staging environment.

- Install fixes and support patches as follows:

Product Release	Action
9.9 and later	Install fixes and support patches from Command Central. First create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.
9.8	Install fixes and support patches from Command Central. First create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.

Product Release	Action
	<p>Note: If you are creating a standalone installation (as opposed to a software stack), install SPM 9.8 Fix 4 or later on the Platform Manager for the products.</p>
9.7	<p>Install fixes from Command Central. First create image repositories that contain the fixes to install using the instructions in this help.</p> <p>Note: If you are creating a standalone installation (as opposed to a software stack), install SPM 9.7 Fix 6 or later on the Platform Manager for the products.</p>
9.6 and earlier	Use Software AG Update Manager to install fixes and support patches.

- If you encounter problems after installing a fix or support patch, contact Software AG Global Support.

Before Creating Database Components

Many Software AG products require an external RDBMS. For those products, you create *database components*. A database component is a grouping of database objects used by one or more products.

You create database components using Command Central and the Database Component Configurator. Install the Database Component Configurator on a machine that has access to your database server and then install the latest fixes on Database Component Configurator.

For information on...	See...
Supported RDBMSs	<i>System Requirements for Software AG Products</i>
<ul style="list-style-type: none"> ■ Database connection information. ■ Database components that are required by each product. ■ Database drivers. ■ Tasks to perform before creating database components. 	<i>Installing Software AG Products</i>

V Creating or Upgrading Standalone Product Installations

■ Understanding Standalone Product Installations	46
■ Install and Configure Platform Manager	46
■ Install Products	49
■ Create Database Components and Connect to Products	50
■ Install Assets	50
■ Upgrade Products	50

Understanding Standalone Product Installations

You can use Command Central to create and upgrade product installations, product instances, and database components within environments.

The set of environments that are managed by a particular Command Central is called a *landscape*.

You can perform these tasks from the Command Central GUI or by running Command Central commands.

You can automate installation and upgrade tasks for product environments by developing *composite templates* that define an environment using domain specific language (CC DSL) and applying the templates using commands. Templates can do the following:

- Provision new development, test, and production environments on empty host machines.
- Upgrade and migrate environments to the latest product releases.

Note: The term "instance" as used in this section includes both runtime instances and runtime instance components. The latter are independent modules that run within a runtime instance but have their own configurable elements. For example, Task Engine is a runtime instance component on My webMethods Server.

Install and Configure Platform Manager

About Installing Platform Manager

If the target machines are not configured for remote access, you download the Command Central bootstrapper to the target machine and use the bootstrapper to install Platform Manager.

If the target machines are configured for remote access (typically the case for UNIX systems and for Windows systems equipped with OpenSSH), you install Platform Manager from Command Central. Command Central uses the SSH user to connect to the target machine.

If Command Central is in a development environment, you can create a product installation on the Command Central host machine. In this case, you must first install Platform Manager into the directory into which you plan to install the products. The product installation directory must be a different directory than the Command Central installation directory.

Install Platform Manager Using the Command Central Bootstrapper

1. Log on to the target machine as the installation user you created when you installed Command Central.
2. Open your installation email from Software AG and follow the instructions to download the Command Central bootstrapper for the target operating system. If you are installing on a Windows system, extract the ZIP archive into the target installation directory.
3. Open a command window as Administrator for Windows or a shell window for UNIX and run the .bat file or .sh script with the arguments in the table below.

Argument	Value	Default
-D <i>component</i>	Specify SPM.	
-H <i>host_name</i>	DNS name or IP address for Command Central to use to connect to Platform Manager after installation.	Machine on which you are running the bootstrapper
-d <i>path</i>	Full path to the directory in which to install Platform Manager.	Directory from which you are running the bootstrapper
-s <i>port_number</i>	HTTP port to use for Platform Manager.	9082
-s <i>port_number</i>	HTTPS port to use for Platform Manager.	9083
-p <i>password</i>	Password for the Platform Manager Administrator user account. If you are installing in a production environment, Software AG recommends you provide a strong, user-defined password.	manage

Argument	Value	Default
<code>-P</code> <code>root_password</code>	If you are installing on a UNIX system and want to register a daemon for Platform Manager, the sudo password.	

- After installation is complete, Platform Manager is running and ready for use.

Note: If you did not register daemons when you installed Platform Manager but want to do so at a later time, you can run the bootstrapper again with the argument `-P sudo_password`.

Examples

- To install Platform Managers on the hosts `linuxhost1`, `solarishost2`, and `windowshost3`, in the `/opt/softwareag` directory, on HTTP port 9997 and HTTPS port 9998, change the Administrator password to `manage456`, and register a daemon using the sudo password `superuser890`:

```
linuxhost1>./cc-def-9.12-release-lnxamd64.sh -d /opt/softwareag -D SPM
-H linuxhost1.com -s 9997 -S 9998 -p manage456 -P superuser123

solarishost2>./cc-def-9.12-release-solamd64.sh -d /opt/softwareag -D SPM
-H solarishost2.com -s 9997 -S 9998 -p manage456 -P superuser890

windowshost3>cc-def-9.10-release-w64.bat -d C:\softwareag -D SPM
-H windowshost3.com -s 9997 -S 9998 -P manage456
```

Install Platform Manager on Remote Machines

You can install Platform Manager into multiple directories at the same time. For directories that are on the same machine, the Platform Manager are installed sequentially, so that only one installation session is executed at a time. For directories that are on different machines, the Platform Managers are installed at the same time.

- Log on to the Command Central machine as the installation user you created when you installed Command Central.
- Go to **Environments > ALL > Installations**, click , click **Add Installation**, and provide the requested values.

Install Platform Manager on a Command Central Host

- Log on to the Command Central machine as the installation user you created when you installed Command Central.
- Go to **Installations > ALL > Installations**, click , click **Add Installation**, and provide the requested values. Specify a different directory than the Command Central installation directory.

Configure Platform Manager

1. To secure the connection with Command Central, go to **Environments > ALL > Instances > SPM > Configuration > Ports > defaultHTTPS** and provide the requested values in the **Security Configuration** area.
2. Command Central comes with default login modules that perform user authentication. You can use Java Authorization and Authentication Service (JAAS) to create and deploy additional, custom login modules. You would only need custom login modules for Platform Manager if you want to connect directly to Platform Manager from a third-party product, such as a monitoring system. For instructions on creating and deploying custom login modules, see *Software AG Infrastructure Administrator's Guide*.

To use JAAS, go to **Environments > All > Instances > SPM > Platform Manager > Configuration > JAAS Realms**.
3. To set timeouts and validations, go to **Environments > ALL > Instances > SPM > Configuration > Java System Properties** and configure the properties as described.

Install Products

Note: Depending on the number of products you select for installation, the product installation job could take some time to complete.

1. Go to **Environments > ALL > Installations** and click the target installation.
2. On the Products tab, click , click the product repository to use, and select the products to install. Command Central automatically manages dependencies for the products you select. When you select a product to install, Command Central automatically selects additional items that are required by the selected product. Auto-selected items include product-specific configuration, administration, and monitoring plug-ins to Command Central; product-specific and third-party libraries and bundles; and other elements of the infrastructure needed by Software AG products.
3. After installation is complete, install the latest fixes on the installed products using the instructions in this help.
4. If a product you installed supports the creation of multiple instances in the same installation directory (for example, Integration Server), Command Central did not create an instance during installation. Create and configure instances using the instructions in the documentation for that product. You might be prompted to install fixes on the new instances.

If you create the instance on a UNIX machine, some products ask whether to register a UNIX daemon. For those that do not ask this question, see instructions on registering daemons in the installation guide for your products.

5. Software AG requires license keys for some products. Software AG provides license key keys for these products when you first license the product, when you need to replace license keys that are about to expire, or when you need to change your license so you can access different product features.
 - a. Go to **Environments > All > Instances** and click the instance or component that requires the license key.
 - b. Click **Configuration**, click **Licenses**, click the license key, click **Edit**, and click the license key to use.
6. If any product you installed or instance you created has a default password, change that password as soon as possible using the instructions in *Administering Software AG Products Using Command Central*.

Create Database Components and Connect to Products

Create database components using the instructions for the `sagcc exec administration` command and composite templates in this help. Then connect the products to their database components using the instructions in the installation guide for your products.

Install Assets

See the `sagcc exec provisioning assets` command in this help.

Upgrade Products

This topic must be used with *Upgrading Software AG Products*. To see which products support upgrade using Command Central, see the *Software AG Command Central Feature Support Matrix*. Also see *Administering Software AG Products Using Command Central* for any product-specific migration instructions.

Note: You do not have to upgrade Platform Manager.

If you need to upgrade multiple similar environments, you can set up composite templates that automate the upgrade (see the template instructions in this help). You can reconfigure endpoints such as host names and ports by adding commands after the migration section of the templates. You will still need to perform the manual tasks in *Upgrading Software AG Products* if you cannot script them.

If you need to upgrade only one environment, you can use Command Central for many upgrade tasks, as described below.

1. Read and follow the instructions in the first several chapters of *Upgrading Software AG Products*.
2. Use Command Central instead of Software AG Installer to install the new products.

Note: If Command Central managed the old product installation, the Command Central installation wizard offers an option that can auto-select the same set of products for the new installation.

3. Use Command Central instead of Software AG Update Manager to install the latest fixes on the new products described in *Upgrading Software AG Products*.
4. Use Command Central to shut down Software AG Runtime as instructed in *Upgrading Software AG Products*. Also disable automatic startup for the new products.
5. If Command Central did not manage the old products, use the instructions in *Upgrading Software AG Products* to prepare the old environment. If Command Central managed the old products, prepare the old environment as follows:
 - a. If you are upgrading products from 9.7 or later, use Command Central to install fixes on the old products listed in *Upgrading Software AG Products*.
 - b. Perform the other preparation tasks for the old products described in *Upgrading Software AG Products*. If you are upgrading Integration Server from 9.8 or later, you can pause the Integration Server instance in Command Central to suspend triggers and drain message queues.
 - c. Use Command Central to shut down the old product instances.
6. Prepare for database component migration using the instructions in *Upgrading Software AG Products* and in the topic "[Before Creating Database Components](#)" on [page 44](#) in this help.
7. Use the Command Central `sagcc exec administration` command instead of the instructions in *Upgrading Software AG Products* to migrate database components. The command syntax is shown below. The parameters are the parameters you use when running the Database Component Configurator and are documented in *Installing Software AG Products*.

```
sagcc exec administration product DCC_node_alias DatabaseComponentConfigurator
database migrate [parm1=value1 ] [parm2=value2 ]...
```

In the examples below, the database components are on a SQL Server RDBMS at `jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB`, and the database user is `webmuser` with password `webmpass`.

- To upgrade Integration Server database components from 9.9 to the latest release:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=sqlserver product=IS version=latest
db.username=webmuser db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

- To upgrade the My webMethods Server database component to the latest release:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=sqlserver component=MWS version=latest
db.username=webmuser db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

To check the progress of the migration in Command Central, click **Jobs** on the Command Central web user interface.

8. *Upgrading Software AG Products* provides product-specific migration chapters that describe pre-migration tasks to perform. Perform the tasks for all products you are going to migrate.
9. If you are going to install the new products on different machines than the old products, use Command Central instead of the instructions in *Upgrading Software AG Products* to create ZIP files of the old installations. Command Central automatically creates a ZIP file of each old installation directory. The ZIP file includes all files except .log files. If you want to decrease the size of the ZIP file, *Upgrading Software AG Products* lists the commands that will include only the required files for each product in the ZIP file.
10. *Upgrading Software AG Products* explains how to migrate your products.

Most products offer migration utilities that migrate their configurations, data, and assets, but some do not. To migrate products that do not offer migration utilities, follow the instructions in *Upgrading Software AG Products*.

For products that do offer migration utilities, *Upgrading Software AG Products* explains the actions that each product migration utility takes by default, but also provides instructions for creating customized migrations that can be called by the migration utilities instead.

Use Command Central instead of the instructions in *Upgrading Software AG Products* to run migration utilities. Use the Command Central command `sagcc exec administration`, shown below.

To find a *product_ID* to use in the command, run the command `sagcc list inventory products node_alias`.

If you want to migrate all instances, do not specify the `instanceName` option.

```
sagcc exec administration product product_node_alias
product_ID migration migrate
{srcDir|srcFile}=full_path_to_{old_Software AG_directory|ZIP file}
[instanceName name [,name ,name ...]]
[importFile=full_path_to_migrate.dat]
[-cloneDbURL URL -cloneDbUser user -cloneDbPassword password]
```

To check the progress of the migration, click **Jobs** on the Command Central web user interface.

11. *Upgrading Software AG Products* provides product-specific migration chapters that describe post-migration tasks to perform. Perform the tasks for all products you migrated.
12. Perform the final upgrade tasks in the last chapter in *Upgrading Software AG Products*.

VI Creating Software Stacks

■ Understanding Software Stacks	54
■ Create a Software Stack	55
■ Install Assets	56

Understanding Software Stacks

You can use Command Central to create, monitor, and maintain multiple product installations using bulk operations by provisioning those product installations as software stacks. A software stack is a set of product runtimes and related database components that serve one or more purposes, such as business process management, application integration, or API management. For information on types of software stacks you might create, see *Understanding Software AG Products*.

Note: The new stacks feature is available as an early iteration for you to investigate. Stacks you create cannot yet be modified, updated, or upgraded. Software AG welcomes feedback on this new feature at Software AG TECHCommunity.

Stacks are made up of at least one of each of the layers described below. You create each layer based on micro templates you develop.

Layer Type	Contents
Infrastructure	Platform Manager. The template identifies the physical machines, virtual machines, or Docker containers that host Platform Manager.
Runtime	Instances of one type of runtime. The template specifies instance creation, configuration, and licensing; database component creation; and product-to-database connections. The runtime instances in one stack cannot be used in any other stack.

In development, you can create all layers of a stack on the same machine. In production, you distribute the layers across machines. You can install different types of runtime instances on the same machine, but if you install on multiple machines, the installations must be identical. For example, if you install product A on machines 1 and 2, you can install product B on machines 1 and 2, but you cannot also install product B on machine 3. The set of products, fixes, and configurations, and assets must be consistent across machines.

You create a layer based on micro templates you develop using domain specific language (CC DSL). Micro templates can provision new stacks on empty host machines. They are modular and reusable.

You can create the stacks from the Command Central GUI or by running Command Central commands.

You can easily monitor and maintain the runtime instances in a software stack using bulk operations. You can:

- Monitor the overall status of the stack to see whether all instances are running, some instances are down, or all instances are down.

- Start, stop, or restart all instances in a layer simultaneously.
- Update the configuration and fixes for one instance in a layer and then synchronize all other instances in the layer to match the updated instance.

You can also compare instance configurations, products, and fixes.

The set of software stacks that are managed by a particular Command Central is called a landscape.

Note: The term "instance" as used in this section includes both runtime instances and runtime instance components. The latter are independent modules that run within a runtime instance but have their own configurable elements. For example, Task Engine is a runtime instance component on My webMethods Server.

Create a Software Stack

Note: To delete stacks, see the `sagcc delete stacks` command in this help. You cannot deprovision stacks.

1. If you must manually install Platform Manager on target nodes (for example, because the user account created to install products does not allow remote access from Command Central), use the same password for every Platform Manager that will be included in the stack you are going to create.
2. Create aliases that specify credentials you will need for the stack. Go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials**, click , and follow the steps below.
 - If you manually installed Platform Manager on target nodes, or if you are going to create an infrastructure layer that connects to new remote nodes that are already equipped with Platform Manager but that are not yet registered in Command Central, click **User name and password** and create an alias for the Platform Manager credentials.
 - If you are going to create an Infrastructure layer that creates remote nodes, create these aliases:
 - Click **User name and password** and specify the credentials to use as the Administrator user name and password for the Platform Managers that will be installed on the remote nodes.
 - If you are installing on Windows systems, click **User name and password** and specify the credentials for the Windows installation user account you created earlier.
 - If you are installing on UNIX systems, click **User name and password** or **SSH private key** as appropriate and specify the credentials for the UNIX installation user account you created earlier.

3. Develop and register micro templates for layers that install products, install fixes, and create database components. For instructions, see the template instructions in this help. Micro templates are intended to be modular and reusable; when you create an instance of a layer for a stack, you will be able to override template parameters and customize them for that stack.
4. Create definitions of layers to use in stacks using the methods below. A software stack must have at least one Infrastructure layer and at least one Runtime layer. When you create a layer instance for a stack, the layer definition will show micro template values by default, and you will be able to override these values.
 - Use basic layer definitions provided by Software AG. You can use these definitions as they are or you can customize them. Basic layer definitions are available at <https://github.com/SoftwareAG/sagdevops-templates>.
 - Go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Layer Definitions**, click , and provide the requested values.
5. Go to **Views > Stacks** and click . Add the stack and then add layers. At the end of the wizard, provisioning begins automatically.
6. Software AG requires license keys for some products. Software AG provides license key keys for these products when you first license the product, when you need to replace license keys that are about to expire, or when you need to change your license so you can access different product features.
 - a. Go to **Environments > All > Instances** and click the instance or component that requires the license key.
 - b. Click **Configuration**, click **Licenses**, click the license key, click **Edit**, and click the license key to use.
7. If any product you installed or instance you created has a default password, change that password as soon as possible using the instructions in *Administering Software AG Products Using Command Central*.
8. Connect the products to their database components using the instructions in the installation guide for your products.

Install Assets

See the `sagcc exec provisioning assets` command in this help.

VII Creating and Viewing License Metering Reports

■ Understanding License Metering	58
■ Create and View a Snapshot License Report	60
■ Create and View an Aggregated Report	60
■ Create and View a Product Usage Report for Billing	60

Understanding License Metering

Current State of a Landscape

If you want to check the current state of installed product instances in a Command Central landscape, you generate a snapshot report. You can generate a snapshot license report by license key or by license manifest.

A snapshot report by license key contains information about installed product instances and groups them based on license key. The report also counts the server cores and compares their total with the number of license keys. The summary section of the report shows whether the customer landscape matches the available licenses.

A snapshot report by license manifest contains information about installed product instances based on a license manifest file that contains all active licenses for a customer. Software AG issues a separate license manifest file for each location of your organization. The report provides detailed information for each installed product, including the type of product installation, the number of installed and running product instances, the server processor types, and the type of license for the product. The report also counts the overall number of server cores and compares their total with the number of licensed products in the license manifest. The summary section of the report shows whether the customer landscape matches the available licenses, and the number of products that match or do not match the licenses.

Continuous License Metering of a Landscape

If you want to continuously meter installed product instances in a Command Central landscape, you generate an aggregated report. Command Central meters the landscape by active license manifest files hourly and stores the results in a monthly report.

The report contains a summary section for each location of your organization for which a license manifest file has been issued. The summary section shows whether the installed capacity (installed processor cores) for a product matches the licensed capacity (licensed processor cores) for the product. You can also view more product details including the daily status of a product, the number of installed and running product instances, the number of used and licensed cores, and whether the installation type and core class match the licenses. A monthly status summary per product is also available.

Important: Command Central does not delete aggregated reports automatically. The average persistent memory usage in Command Central for an aggregated report is 100-300 kilobytes per product.

Command Central automatically starts continuous metering report aggregation. You cannot start or stop report aggregation.

Monthly Metering of a Landscape

The pricing contract and licenses for some products are based on usage. If you want to meter installed product instances in a Command Central landscape monthly based on usage, you generate a *product usage report for billing*. This report is an excerpt of the aggregated report and is used by Software AG for billing. There are two types of usage-based licenses.

License	Charges are based on...	Pricing Model	Report Data
Base and peak	Product components such as processor cores.	A base number is defined for each type of component and usage is charged when those numbers are exceeded (peak).	Peak number of components for each product.
Total usage	Metrics such as events or connections.	All usage is charged, or a base number is defined for each metric and usage is charged when those numbers are exceeded.	Total usage or usage above the defined base usage, depending on the pricing model.

The report also contains information such as the daily status of a product, the number of installed and running product instances, and the number of used and licensed components.

All usage-based pricing installations have a contract item identifier that is a unique ID for a contract item in a license manifest. The contract item identifier consists of a location ID, contract ID, contract item ID, and bundle component ID. The product usage report for billing gathers information from all active license manifest files that contain a usage-based license.

If Command Central can find the license from a product license key in an active license manifest file, it automatically assigns the license to the product instance. If Command Central cannot find the license key in an active license manifest file, you must assign the license to the product instance yourself. Command Central automatically starts continuous metering and monthly report aggregation. You cannot start or stop report aggregation.

Create and View a Snapshot License Report

1. If you want to create a snapshot license report by license manifest, do the following:
 - a. Contact your Software AG sales person to obtain a manifest file, and then save the manifest file.
 - b. Go to **Licensing > Manifests**, click , and provide the requested values.
2. Go to **Licensing > Reports**, click , and click the type of snapshot license report to create.
3. To view the contents of the report, click  and then click a format.

Create and View an Aggregated Report

1. Go to **Licensing > Reports**, click  and then click **Start Report Aggregation**.
2. To view the contents of the report, click  and then click a format.

Create and View a Product Usage Report for Billing

1. Go to **Environments > ALL > Instances**, click the instance or component to which to assign a usage-based license.
2. On the Overview tab, in the **License** field, click .
3. In the **License Contract Item** field, click the license to assign.
4. To view the contents of the report, go to **Licensing > Reports**, click  and then click a format.

VIII Tuning and Troubleshooting

- Configure the Command Central and Platform Manager JVMs 62
- Troubleshooting 63

Configure the Command Central and Platform Manager JVMs

When you installed Command Central and Platform Manager, a JDK was installed with them. Command Central monitors the JVM for various fault conditions and takes a specified action when a fault occurs, as shown below. Do not change the settings for any of these features unless specifically asked to do so by Software AG.

Feature	Default
Detect a non-operational (hung) JVM. After the JVM starts, Command Central pings it periodically. If the JVM does not respond within a specified interval, Command Central assumes the JVM has stopped functioning and restarts it.	Enabled
Detect thread deadlocks in the JVM. A thread deadlock occurs when two or more threads try to lock resources in a manner that causes all threads to wait indefinitely. Command Central can monitor the JVM for a deadlock condition and take a specified action (for example, restarting the JVM) when the condition occurs.	Enabled
Detect specified messages in the console output. Command Central can monitor the console output and take a specified action when a given text string appears. This feature is often used to watch for out-of-memory messages.	Disabled

Command Central requires a JDK, but Platform Manager can use a JRE. If necessary, you can specify a different JDK or JRE by modifying the `wrapper.java.command` property in the Command Central and Platform Manager `custom_wrapper.conf` files in the *Software AG_directory*/profiles/CC or /SPM directory, respectively. After editing and saving the file, restart the product.

Important: If you specify a different JDK or JRE, do not remove the JDK that Software AG Installer installed with Command Central and Platform Manager. It is required to run the Uninstaller.

You can modify the Java heap size for Command Central and Platform Manager, and you can modify or add JVM options.

1. Go to **Environments > ALL > Instances > {CCE|SPM} > Configuration**.
2. By default, the initial size of the Java heap for Command Central and Platform Manager is 32MB and the maximum size is 512MB. To modify the heap size, click **Memory** and then click **Edit**.
3. To modify other JVM options, or add new JVM options, click **JVM Options** and then click **Edit**.

Troubleshooting

During file download, activity at times appears to stop completely

Most download issues are caused by interference from a security appliance such as a virus scanner. Ask your network administrator whether he can make an adjustment to allow the download to work properly. Ask the network administrator to check the security settings for your proxy or firewall; they might be incompatible with Command Central. If so, ask your IT department for temporary access to a port outside the firewall to download the files. Product jar files and product files downloaded from Empower are verified using SHA256 checksums.

Bootstrapper fails with error indicating a package error (for example, "installer.jar not found")

If you download the Command Central bootstrapper and then transfer it to another machine, the Install Command Central topic explains that you must set the transfer tool you are using to binary mode. If you see the error noted above, the bootstrapper was corrupted because text transfer mode was used instead. Go to the Empower Product Support website from which you downloaded the bootstrapper and download the SHA256 file for that bootstrapper. Calculate the SHA256 checksum for the bootstrapper and compare it to the checksum value in the downloaded SHA256 file. If the values do not match, delete the corrupted bootstrapper, re-download it, and re-transfer it using binary mode.

Expected product-specific features are not available

Product-specific features in Command Central exist in the form of plug-ins to Platform Manager. To check whether the product plugin is installed and active, go to **Environments > ALL > Instances > SPM > Administration > Diagnostics**. If the missing plug-in is not installed, install it. If the missing plug-in is installed, restart Platform Manager.

Cannot connect to repositories due to invalid credentials

Make sure you have typed your credentials correctly. If you have, go to the Empower Product Support website and try to log in. If you cannot log in, contact your Software AG system administrator.

If you can log in, go to **Download Products < Software Downloads < Software Download Center (SDC)** and click **Download**. If you cannot see the next page, you do not have permission to download products, and should contact your Software AG system administrator. If you can see the next page, you might be experiencing a network connectivity issue and should contact your network administrator.

View and download logs for instances or components

Go to **Environments > ALL**, click the instance or component whose logs to view, and click the Logs tab. To download logs:

Download	Action
One log	Click  for that log.

Download	Action
Multiple logs	Hold down the Shift or Ctrl key and select the log rows. Click  and then click Download selected logs .
All logs	Click Download selected logs without selecting any log.

Edit Command Central or Platform Manager log configurations

Important: Make a backup of log configuration files before editing.

If you are on the machine that hosts the Command Central whose log configurations to edit, go to *Software AG_directory* \profiles\CCE\configuration\logging directory and edit the log_config.xml file. If you are not on that machine, use the command line interface to access the machine. Get the configuration data by running this command:

```
Software AG_directory\CommandCentral\client\bin>sagcc get configuration
data local OSGI-CCE COMMON-LOG-log_config.xml -o log.xml
```

Edit the retrieved log.xml file, then update the configuration using the edited file as input by running this command:

```
Software AG_directory\CommandCentral\client\bin>sagcc update configuration
data local OSGI-CCE COMMON-LOG-log_config.xml -i log.xml
```

If you are on the machine that hosts the Platform Manager whose log configurations to edit, go to *Software AG_directory* \profiles\SPM\configuration\logging directory and edit the spmlog.xml file. If you are not on that machine, use the command line interface to access the machine. Get the configuration data by running this command:

```
Software AG_directory\PlatformManager\client\bin>sagcc get configuration
data node_alias OSGI-SPM COMMON-LOG-log_config.xml -o log.xml
```

Edit the retrieved log.xml file, then update the configuration using the edited file as input by running this command:

```
Software AG_directory\PlatformManager\client\bin>sagcc update configuration
data node_alias OSGI-CCE COMMON-LOG-log_config.xml -i log.xml
```

Cannot create mirror repository because of network issues

Address the network problem and try again to create the mirror repository.

Command Central cannot connect to Platform Managers

If Command Central cannot connect to a Platform Manager, make sure the Platform Manager is running, is listening on the port specified during Platform Manager installation, and your firewall allows incoming connections to Platform Manager and outgoing connections from Command Central. If this does not address the problem, go to **Environments > All > Instances > CCE > Configuration > Proxy** and configure a proxy for Command Central. You might need to exclude Platform Manager hosts and domains.

IX Automation and Template-based Provisioning

■ Using Composite Templates	67
■ Developing Composite Templates	119
■ Using Micro Templates	127

1 Using Composite Templates

■ About Automated Provisioning of Environments	68
■ What is a Composite Template?	68
■ Understanding the Composite Template Definition	70
■ Format of COMMON-* Configuration Types in the Template Definition	97
■ Summary of Sample Composite Templates	99
■ Using the is-layer Sample Template	103
■ How Command Central Processes a Composite Template	111
■ Validating a Composite Template	114
■ Creating a New Environment Using a Composite Template	115
■ Updating a Provisioned Environment Using the Same Composite Template	116
■ Monitoring the Status of a Composite Template	117
■ Correcting a Failed Composite Template Apply Operation	117

About Automated Provisioning of Environments

When you work with large Software AG product environments, which have high scaling requirements, you can use the scripting functions and composite templates provided by Command Central to automate installing, configuring, or upgrading the environments. Based on the release version of the environment that you want to provision or upgrade, use the following functions of Command Central to automate operations:

- **Scripting with CLI commands to administer, monitor, and configure environments of release 9.6 or higher.**

You can use the CLI commands to write scripts to automate configuration and administration operations in an environment, such as stopping or restarting a large number of run-time components and updating configuration settings.

- **Composite templates to provision new environments of release 9.8 or higher.**

With the composite templates you can automate creating a whole new Software AG product infrastructure by applying one composite template on a large number of nodes to create new product installations. In a Command Central composite template, you can describe all elements of the product environments, such as products, instances, fixes, and configuration. The composite template also groups the products into functional layers, and lists the configuration settings and database details for each product instance. You store composite templates as zip archives, which you include in a single CLI command or script to create new installations on remote machines. You select which environments to install on which nodes by either creating a map in the template or adding parameters in the CLI-based script. You should put composite templates under version control and test them in the same way you test an application.

- **Composite templates to provision and upgrade environments of release 9.8 and higher.**

You can also use composite templates to automate the upgrade process for Software AG product environments. When applying a composite template with the `apply composite template` CLI command, you just add a parameter that indicates that Command Central must migrate the environments defined in the template. You can use the default migration flow supported by Command Central, or customize the migration settings in the template definition. For example, you can specify whether to create an archive of the old installation directory, or whether to migrate to the same or a different host.

What is a Composite Template?

With Command Central, you can use composite templates to create new environments and update or upgrade existing environments.

At the core of a composite template is the template definition. The template definition describes the target state of the environment that you want to install or modify, and

Command Central calculates how to achieve the target state, based on the data in the composite template. The template definition is designed with the typical Software AG product environments in mind and takes into consideration the requirements specific for different types of environments, such as which components should be installed and how the components communicate with the system infrastructure or with each other.

The template definition can also include shell actions executed at different phases during the template application process, which enable Command Central to automate an even wider range of provisioning operations.

A composite template is very similar to an application and the concept of infrastructure as code in that it goes through several development stages:

1. Designing and creating

You design the state of the environment that you want to create in the composite template definition by defining a set of environment properties for which you can specify values or use the default values provided in the template. For example, you can specify which products to include in a composite template or the number of nodes on which to apply the template. You also include the following elements in the composite template definition:

- Define repositories, layers, templates, installations, and provisioning maps.
- Specify actions on demand that Command Central executes when applying the composite template, for example migrate a product that does not have a migration utility.
- Reference repositories and external files, such as license key, product, or fix image files.

2. Packaging

If the composite template that you designed does not include references to external files, it is defined in a single YAML template definition file. If the composite template includes references to external files, it is packaged in a zip archive, which consists of a YAML template definition file, named `template.yaml`, and the external files referenced in the template definition.

You can find sample composite templates for different product environments here: <https://github.com/SoftwareAG/sagdevops-9.x-template-samples>

3. Importing and applying

After you create a composite template, you should store it in a version control system and test or modify the template as required. To use the composite template, you must import it in Command Central and apply the template on the target machines. You import and apply composite templates using the composite templates commands of the Command Central command line interface (CLI). You can determine how you want to apply the template using the `apply composite template` command arguments, for example with the `environment.type` command argument you can select which of the environments defined in the template to apply on the target hosts. For more information about how to use the composite template

commands, see ["sagcc exec templates composite import" on page 467](#) and ["sagcc exec templates composite apply" on page 464](#).

Understanding the Composite Template Definition

A composite template uses a custom domain-specific language (DSL) and a template definition in the YAML format. For more information about the YAML format, see the [YAML documentation](#). The YAML template definition file is always named "template.yaml" and located at the root level in the composite template zip archive.

Each composite template is defined with a template alias that is unique for the Command Central environment. The only valid characters in a template alias are ASCII characters, numbers, underscore (`_`), dot (`.`), and a hyphen (`-`).

The template declaration in the template.yaml file consists of the template alias, the DSL version, and an optional description and version fields. The description is short and usually states the purpose of the template. You can optionally specify a version parameter, based on user-defined versions of the updates to the template definition file. For example:

```
alias: templatel
description: |
  This is a basic template
  with a description.
dslVersion: 1.1
version: 0.1
```

Important: Beginning with Command Central 10.2, the version of the DSL is 1.1. Note that some of the instructions and parameters included in DSL version 1.1 are not supported by the earlier versions of the DSL. Command Central 10.1 and lower uses DSL version 1.0.

The following topics describe the syntax and structure of a composite template definition. See also the [Reference Template DSL Definition](#) topic in the Command Central online help.

Environments

Environment properties are defined as input parameters under the `environments` section in the composite template definition. The environments section usually includes `default` and `environment.type` sections.

environments/default

In the default section, you can specify the input parameters that apply to all environments defined in a composite template.

If an input parameter has a default value, you can use the default value or specify a user-defined value. If an input parameter does not have a default value, you must specify a user-defined value when applying the template. Use the following format to specify a value for a parameter:

```
paramName: paramValue
```

Parameter values can include references to other existing parameters with the following syntax:

```
paramName: ${otherParamName}
```

Example

In the following example, “param1” has a default value that you can use or replace. “param2” is a required parameter and you must provide a value for this parameter when applying the composite template. “param3” uses the values of “param1” and “param2” by reference.

```
environments:
  default:
    param1: default
    param2: ${}
    param3: ${param1},${param2}
```

environments/environment.type

You can specify a type for each environment defined in a composite template in the implicit *environment.type* parameter. In the *environment.type* sections, you can either introduce new parameters (with default or custom values), or specify new values for parameters defined in the *environments/default* section.

Use the *environment.type* sections when you want to customize the list of input parameters and their values to fit the requirements a particular environment type. The parameters and values you specify in an *environment.type* section apply only for that environment.

Note that when you specify a default value for a parameter in the *environments/default* section, the environment type sections use that default value, unless you specify a different default value for the same parameter in an *environment.type* section.

Example

In the following example, the environment with type "dev" uses "B" as the custom value of “param2”. The environment of type "cluster" uses "default1" as the default value for “param1”, and “param3” uses the value "C".

```
environments:
  default:
    param1: default
    param2: A
    param3: ${param1}
  dev:
    param2: B
  cluster:
    param1: default1
    param3: C
```

How Command Central Processes Input Parameters

The values provided for the input parameters are resolved in the following order:

1. The values from the `environments/default` section if this section exists
2. The values from the `environments/environment.type` sections
3. The user-defined values
4. If an input parameter does not have a value, Command Central returns an error.

Example:

```
environments:
  default
    param1:defaultValue
    param2:
    param3: ${param1},${param2}
    param4: anotherValue
  envType1:
    param2:someValue
  envType2:
    param1: v1
    param3: v3
```

The values in this example environment definition are resolved as described in the following table:

Parameters	Resolved Parameter Values
no parameters	Error. Missing parameter values. Provide values for the parameters.
param2=foo	param1=defaultValue param2=foo param3=defaultValue,foo param4=anotherValue
<i>environment.type</i> =envType1	param1=defaultValue param2=someValue param3=defaultValue,someValue param4=anotherValue
<i>environment.type</i> =envType2 param2=foo param4=\${param2}	param1=v1 param2=foo param3=v3 param4=foo

Parameters	Resolved Parameter Values
<code>environment.type=bar</code>	<p>Error. The template does not include an <code>environment.type</code> with value “bar”.</p> <p>Either provide a valid value for <code>environment.type</code> or leave empty to use the global default values.</p>
<code>param2=foo</code> <code>param5=bar</code>	<code>param1=defaultValue</code> <code>param2=foo</code> <code>param3=defaultValue,foo</code> <code>param4=anotherValue</code> <code>param5=bar</code>
	<p>Note: The value for <code>param5</code> is not used in the template and is ignored.</p>

Layers

A layer in a composite template is a collection of instances that have the same logical role. For example, an Integration Server cluster is a layer.

The `layers` section of the composite template contains the definition of the layers that will be installed on a set of nodes. A composite template should include at least one layer. Each layer is identified by a unique alias name and includes an ordered list of inline templates. The inline templates are applied on all nodes that map to the layer, following the exact order of the list. The inline templates are defined in line, in the same composite template that contains the layer definition.

Note: If the `templates` list is empty, Command Central does not apply any inline templates to the nodes that map to this layer.

The `layers` section includes an optional `default` subsection in which you can specify a product or fix repository from which to install the products or fixes for all layers defined in the composite template. You can also define a product or fix repository from which to install products or fixes for each layer. When a product or fix repository is defined for each individual layer, Command Central installs products and fixes from the layer specific repositories and ignores the repository details provided in the `default` subsection.

You can add an optional `description` for each layer.

The templates for each layer are listed in the `templates` section.

Example

In the following example, the composite template includes two layers with alias names “layer1” and “esb”. The inline templates under “layer1” and “esb” are applied on the nodes that each layer maps to, but both layers use the product and fix repositories specified in the `default` section:

```
layers:
  default:
    productRepo: webMethods-9.10
    fixRepo: Empower
  layer1:
    templates:
      - myTemplate2
  esb:
    description: generic Integration Server layer
    templates:
      - is-server
      - is-config
```

Templates

You define inline templates in the `templates` section. The templates included in this section are identified by a unique template alias. In the `templates` section, you can include the following optional sections:

products

A list of IDs for the products that you want to install. You can list the product IDs using the [sagcc list repository products content](#) command or by selecting the **Show ID Column** check box when viewing the contents of a product repository in the web user interface.

Example syntax

```
products:
  productID:
    instanceName:
      ${node.host}: value
      ${node.alias}: value
      ${src.node.host}: value
    configuration:
      runtimeComponentID:
        CONFIG-TYPE:
          CONFIG-TYPE-instancel:
            config_setting1: value
            config_setting2: value
```

You can also use a reference variable for `instanceName`:

```
${instance.name}:
```

Note that when a product does not support multiple instances the instance name is default:

In the `templates/templateId/products/productId/instanceName` sections, you can include the following context parameters:

- `${node.host}`:
Resolves to the host of the node on which the template is provisioned.
- `${node.alias}`:
Resolves to the alias of the node on which the template is provisioned.
- `${src.node.host}`:
When `environment.mode=MIGRATION`, resolves to the host of the source node for the migration. For details about the migration settings you can specify for a product instance defined under `instanceName`, see ["Migration Settings for Product Instances" on page 84](#).

Based on the type of data that each configuration setting supports, you can specify the values in the configuration settings parameters in one of the following formats:

- YAML
- JSON
- XML
- multi-line plain text
- variable

For details about the supported formats for each COMMON-* configuration type, see ["Format of COMMON-* Configuration Types in the Template Definition" on page 97](#).

fixes and support patches

Use the `fixes:` section to specify a list of IDs for the fixes that you want to install. Command Central installs the latest available version of each fix and its dependencies from the fix repository defined in the template. You can find the fix IDs and the exact version number of a fix using the ["sagcc list repository fixes content" on page 422](#) or by selecting the **Show Fix Name Column** check box when viewing the contents of a fix repository in the web user interface. For example:

```
fixes: [wMFix.Product1Fix, wMFixProduct2Fix]
```

You can include the exact version number of a fix in the following format `wMFix.productFixID_major.minor.service.fix-build`. For example:

```
fixes: [wMFix.Product1Fix_10.2.0.0001-0001, wMFix.Product2Fix_10.2.0.0003-0123]
```

If the `fixes:` list is empty, Command Central does not install any fixes:

```
fixes: []
```

The `fixes:` section functions differently, based on the version of the template DSL.

Template DSL version 1.0

With template DSL version 1.0, support patches are listed in the `fixes:` section, for example:

```
fixes: [wMFix.Product1Fix, wMFix.Product2Fix, patchKey1, patchKey2]
```

If you set the `fixes:` section to `ALL`, Command Central installs all the latest fixes required for the products in the installation:

```
fixes: ALL
```

If you include the `patches:` section, the template validation fails.

The following example shows how the `fixes:` section functions with template DSL version 1.0:

- If "product1" and "product2" are not installed, Command Central installs "product1" and "product2" from the "repoProducts" repository.
- Command Central installs "wMFix.fix1_v1", "wMFix.fix2_v3", "cdkey1", and "cdkey2" with their dependencies, in the order in which they are listed, from the "repoFixes" repository . Command Central uninstall automatically all support patches that prevent installing the fixes from the list.
- Command Central creates "product2_Instance" for "product2 " (if the instance does not exist).
- Command Central re-installs the fixes and patches required for the product2 instance (the fixes are installed before the patches). However, Command Central cannot uninstall the support patches that prevent re-installing the official fixes required for the product2 instance, which might cause the template execution to fail.

```
alias: sagTemplate
dslVersion: "1.0"
layers:
  layer1:
    productRepo: repoProducts
    fixRepo: repoFixes
    templates: template1
templates:
  template1:
    fixes: [wMFix.fix1_v1, wMFix.fix2_v3, cdkey1, cdkey2]
    products:
      product1:
      product2:
        product2_Instance:
```

Template DSL version 1.1

Important: Beginning with template DSL version 1.1, you cannot list support patches in the `fixes:` section.

If you specify `ALL`, Command Central installs all the latest fixes required for the products defined in the inline template:

```
fixes: ALL
```

Use the `patches:` section to specify a list of support patches that you want to install in addition to the installed fixes. Software AG strongly recommends that you do not install support patches in a production environment.

```
patches: [patchKey1, patchKey2]    #Support patches get installed
                                   #in the order in which they are listed.
```

If the `patches:` list is empty, Command Central does not install support patches.

```
patches: []
```

Command Central automatically uninstalls all existing support patches that prevent the installation of fixes or support patches defined in the `fixes:` and `patches:` sections. Any other support patches that do not get in the way of installing the fixes and support patches listed in the inline template, remain unchanged.

If you specify a list of `fixes:` and a list of `patches:`, the fixes get installed before the support patches.

Examples:

- In the following example:
 - If "product1" and "product2" are not installed, Command Central installs "product1" and "product2" from the "repoProducts" repository.
 - Command Central installs "wMFix.fix1_v1", "wMFix.fix2_v3", "cdkey1", and "cdkey2" with their dependencies, in the order in which they are listed, from the "repoFixes" repository. Command Central uninstalls automatically all support patches that prevent installing the fixes from the list.
 - Command Central creates "product2_Instance" for "product2" (if the instance does not exist).
 - Command Central re-installs the fixes and patches required for the product2 instance (the fixes are installed before the patches). Command Central also automatically uninstalls all support patches that prevent re-installing the official fixes required for the product2 instance.

```
alias: sagTemplate
dslVersion: "1.1"
layers:
  layer1:
    productRepo: repoProducts
    fixRepo: repoFixes
    templates: template1
templates:
  template1:
    fixes: [wMFix.fix1_v1, wMFix.fix2_v3]
    patches: [cdkey1, cdkey2]
    products:
      product1:
      product2:
        product2_Instance:
```

- In the following example:
 - If "product1" and "product2" are not installed, Command Central installs "product1" and "product2" from the "repoProducts" repository.
 - Command Central installs only the fixes in the "repoFixes" repository that are required for "product1" and "product2" (with all their dependencies), but does not install fixes for any other products that already exist in the same installation with "product1" and "product2".
 - Command Central creates "product2_Instance" for "product2" (if the instance does not exist).

- Command Central re-installs the fixes and patches required for the product2 instance (the fixes are installed before the patches). Command Central also automatically uninstalls all support patches that prevent re-installing the official fixes required for the product2 instance.

```
alias: sagTemplate
dslVersion: "1.1"
layers:
  layer1:
    productRepo: repoProducts
    fixRepo: repoFixes
    templates: template1
templates:
  template1:
    fixes: ALL
    patches: []
    products:
      product1:
      product2:
        product2_Instance:
```

When the template is executed for a second time with the same fix repository, if "repoFixes" does not include new fixes, the template does not make any changes to the installed support patches and leaves them as is. If "repoFixes" contains a new set of official fixes for "product1" and "product2" and their dependencies, the template installs this new set of fixes and uninstalls the support patches that prevent the installation of the fixes.

files

A map of files that Command Central will copy to the target installation. For example:

```
target/file1: "src/file"
```

where `target` is the path to the directory to which to copy the file and `src` is the path to the source directory from which to get the file. The target path is always relative to the target installation directory and the source path is relative to the root folder of the composite template.

licenses

A map of license key aliases that Command Central uses to copy the license key file assigned to a specific license key alias to the target location. For example:

```
"target/licenseKey1": "${alias}"
```

where `target` is the path to the directory to which to copy the license key file and "alias" is the license key alias. The "\${alias}" value is not case-sensitive and you can use wildcard characters, for example:

```
"target/licenseKey": "*_<productCode>_<majorVersion>.*_${os.platform}"
```

You refer to the license key file by an alias, which you can define in:

- the Licensing view of the Command Central web user interface
- an environment properties file

- the CLI `license-tools keys` commands
- an argument of the `sagcc exec templates composite apply` command

Reference parameters for node aliases

You can refer to a node alias from an inline template in the `templates` section using one of the following parameters by reference:

Note: The parameters in the table are supported with DLS version 1.1.

Parameter	Description
<code>\${node.alias.prefix}</code>	Use only when the mapping policy for node alias is set to <code>INDEX</code> . The prefix of the node alias. When policy <code>HOST</code> or <code>EXISTING</code> , the value of this parameter is empty.
<code>\${nodes[i]}</code>	The node aliases of the infrastructure layer with an index number, for example <code>\${nodes[1]}</code> . The parameter includes all node aliases for the entire environment, not just for the layer in which the inline template gets included.
<code>\${hosts[i]}</code>	The hosts of the infrastructure layer with an index number, for example <code>\${hosts[1]}</code> . The parameter includes all hosts for the entire environment, not just for the layer in which the inline template gets included.
<code>\${nodes.count}</code>	The number of nodes on which to apply the template, for example 3. The value is 1 or higher, although 0 is also a valid value.
<code>\${node.index}</code>	The index number of the node in the entire environment. The value starts at 1 and goes as high as <code>\${nodes.count}</code> .
<code>\${myLayer.nodes[i]}</code>	The node alias of <code>myLayer</code> with a node index number, for example <code>\${um.nodes[1]}</code> .
<code>\${myLayer.hosts[i]}</code>	The node alias of <code>myLayer</code> with the host index number, for example <code>\${um.hosts[1]}</code> .

Parameter	Description
<code>\${myLayer.nodes.count}</code>	The number of nodes in <code>myLayer</code> . The value starts at 1 and goes as high as <code>\${nodes.count}</code> .
<code>\${myLayer.node.index}</code>	The index number of the node in <code>myLayer</code> . The value starts at 1 and goes as high as <code>\${myLayer.nodes.count}</code> .

Migration

You can define an optional `migration` section in which you specify custom options and migration details for an environment. When you do not include a `migration` section in a composite template that you intend to use to migrate an environment, Command Central uses the default migration settings as described in ["How Command Central Processes a Composite Template" on page 111](#). However, when you want to use a composite template for migration, you must always include the ["Provision" on page 86](#) section.

The `migration` section can include one or more of the following sections:

sourceType

Specifies whether to migrate an environment from an existing archive of the source nodes or directly from the source nodes included in the `migration/nodes` section. If you do not include this parameter or set the value to `NODES`, Command Central ignores the `migration/archives` section. If you set the value to `ARCHIVES`, Command Central ignores the `migration/nodes` section. The default is `NODES`.

Syntax:

```
sourceType: [ARCHIVES|NODES]
```

archives

Specifies the location of an existing archive of the source nodes. Note that Command Central ignores this section when `sourceType: NODES` and uses the node details in the `migration/nodes` section instead. By default, Command Central searches for the source archive on the Command Central host, in `CommandCentral_directory/profiles/CCE/data/migration/source/nodeAlias.zip`. If Command Central does not find the archive on its own host, it searches on the target host, in `user.home/migration_source_nodeAlias.zip`.

Syntax:

```
sourceType: ARCHIVES
archives:
  targetPath: path/to/source_archive.zip
```

options

Migration options to prepare the source nodes for migration. Following is a list of the options that you can customize:

■ snapshot

Creates a template with products, fixes, configuration, and files from each source node. The template does not include product instances. The product instances are created by the product migration utilities. The default value for source installations of release 9.6 or higher is `true`.

Syntax:

```
snapshot:  
  execute: {true | false}
```

■ pause

Pauses all source run-time components. The default value is `true`.

Syntax:

```
pause:  
  execute: {true | false}
```

Note: Command Central requests of all run-time components to pause, but a component processes this request only when it supports the pause operation. For example, Integration Server goes into Quiesce mode, instead of pausing.

■ shutdown

Stops all source run-time components. This option also shuts down Platform Manager by running the `shutdown.sh/bat` script remotely through the `remoteAccess` service. You can set the `shutdown` option to `false` only when migrating to a different host on a Unix operating system. For all other types of migration, the default value is `true`.

Important: To execute the shut-down operation successfully, Command Central must access the remote operating system through an SSH connection.

Syntax:

```
shutdown:  
  execute: {true | false}
```

■ backup

Creates a backup of the source installation directory on the same host machine and transfers the backup archive to Command Central. Note that this step is skipped automatically when migrating on the same host. The backup operation is executed only when you set `execute=true`, which is the default value. The output of the backup operation is an archive.

Important: To execute the back-up operation successfully, Command Central must access the remote operating system through an SSH connection.

When migrating to a different host, in the `sourcePath` parameter you specify the location in which to create an archive of the installation on the source node and in the `targetPath` parameter - the location to which to transfer the archive on the target node.

If you do not specify a value for `sourcePath` or `targetPath`, Command Central uses as default location:

```
user.home /nodeAlias
```

where *user.home* is the home directory of the user on the source or target machine respectively and *nodeAlias* is the alias of the source installation.

To specify which files to exclude from the backup, use the `excludes` option.

You can select which file types to exclude from the source archive in a more granular way, using the optional `patterns` and `except` parameters. For example, you can use these parameters when you want to reduce the size of the source archive, but still include required files when migrating to a new host. The `patterns` parameter specifies which file types to exclude from the source archive. The default value for `patterns` is `["*.log"]`, which means that by default only log files are excluded from the source archive. The `except` parameter includes a list of file names that are exceptions. The files listed in this parameter are included in the source archive, even when their file type is listed in the `patterns` parameter. For example, in the following template snippet, all jar files will get excluded from the source archive, except for the "wm-caf-server.jar" file:

```
migration:
  options:
    backup:
      excludes:
        patterns: ["*.log", "*.jar", "*.zip"]
        except: ["wm-caf-server.jar"]
```

Syntax:

```
backup:
  execute: {true | false}
  sourcePath: user.home /nodeAlias
  targetPath: user.home /nodeAlias
  excludes:
    patterns: ["*.log", "*.jar", "*.zip"]
    except: [list-of-file-names-excepted-from-patterns]
```

■ rename

Renames the source installation directory to `migration_source_installDir`, where `installDir` is the name of the source installation directory. The rename operation is executed only when migrating on the same host and installation directory (overinstall) or if you set `execute=true`.

Syntax:

```
rename:
  execute: {true | false}
```

```
targetPath: <the path goes here>
migration_source_installDir
```

source

A list of the source nodes that host each layer. The source nodes are defined in `source/environment.type/layerAlias` sections. For example, the source nodes for an “is” layer in a “cluster” environment are defined in the `migration/source/cluster/is` section.

nodes

Details about the source nodes, such as host, port, and bootstrap information. Note that Command Central ignores this section when `sourceType: ARCHIVES` and uses the source node archive specified in the `migration/archives` section instead.

Example

In the following composite template snippet, the “cluster” environment has a single “is” layer. The target migration nodes are identified from the “provision/cluster/is” section and the source migration nodes from the “migration/source/cluster/is” section. The source and target nodes use the same “is1” and “is2” hosts. Because the “migration/nodes” section does not have any properties, the source nodes map to the Platform Manager port “8093” and the “/opt/sag/test” installation directory, defined in the `nodes/default` section. The migrate operation creates a template with the products, fixes, configuration, and files for each source node, but does not create a backup archive. All run-time components, including Platform Manager are shut down before starting the migration process and the source installation directory is renamed to “migration_source_cc910”.

```
environments:
  default:
    cc.installer: ${}
    install.dir: /opt/sag/test
    db.url: "jdbc:wm:sqlserver://dbhost1:1433;databaseName=db1"
    db.username: user1
    db.password: pass1
  cluster:
    is.hosts: [is1,is2]

layers:
  default:
    productRepo: webMethods-9.10-mirror
    fixRepo: Empower-9.10-mirror
  is:
    templates: [spm-tuneup,is-server]

templates:
  spm-tuneup:
    fixes: [wMFix.SPM.Core,wMFix.PIEspm]
  is-server:
    fixes: [wMFix.integrationServer.Core]
    products:
      integrationServer:
        is_inst1:
          primary.port: 5555
          db.url: ${db.url}
          db-username: ${db.username}
```

```

        db-password: ${db.password}
migration:
  options:
    snapshot:
      execute: true
    pause:
      execute: true
    backup:
      execute: true
      excludes: ["*.jar","*.log"]
    shutdown:
      execute: true
    rename:
      targetPath: /opt/sag/cc910
  source:
    cluster:
      is: [is1,is2]
  nodes:
provision:
  cluster:
    is: [is1,is2]
nodes:
  default:
    default:
      port: 8093
      secure: true
      bootstrapInfo:
        installer: ${cc.installer}
        installDir: ${install.dir}

```

Migration Settings for Product Instances

You can specify migration settings for a product instance defined in `templates/templateId/products/productId/instanceName`. These migration settings apply only for the product instance for which they are defined.

You can add the following parameters for a product instance and specify the values for the parameters in an environment properties file or as arguments of the `sagcc apply composite templates` command:

```

templates:
  templateId:
    products:
      productId:
        instanceName:
          ${environment.mode}:
          ${migration.type}:
          ${src.node.host}:

```

Parameter	Description
<code>environment.mode</code>	Optional. Indicates whether to install or migrate the product instance. The values are: <ul style="list-style-type: none"> ■ PROVISIONING - installs the product instance

Parameter	Description
	<ul style="list-style-type: none"> ■ MIGRATION - migrates the product instance
<code>migration.type</code>	<p>Optional. Specifies how to migrate the product instance. The values are:</p> <ul style="list-style-type: none"> ■ OVERINSTALL - migrate on the same host and into the same installation directory ■ SAME_HOST - migrate on the same host, but into a different installation directory ■ CROSS_HOST - migrate to a different host ■ NONE - migrate without using layers
<code>src.node.host</code>	<p>Optional. The value of this parameter resolves to the hostname of the source Platform Manager installation. When migrating to a different host, this value is different from the host name of the target installation. For all other supported types of migration, this value is the same as the target hostname.</p>

You can also include the migration settings in properties files, named in the following format:

```
migrate_srcVersion -destVersion _sbs.dat
```

for example, `migrate_9.12.0-10.1.0_sbs.dat`

Add the .dat files in the composite template archive in the `migration/productId` directory.

If an inline template defined in the composite template includes only the product, but does not define any instances for that product, the settings in the .dat file under the directory with this product ID are applied for the product, but not for specific instances of the product. If the inline template includes the product and its instances, the settings in the .dat file for this product ID are applied for all instances defined for that product in the inline template.

Important: Note that on the Platform Manager side, the custom migration settings will be ignored if the migration source and target versions are different from the versions specified in the name of the custom migration settings file.

Example

In the following example, the product with ID "MwsProgramFiles" has a single instance with name "mws_inst1". To use custom migration settings for "mws_inst1", create a properties file with name "migrate_9.12.0-10.1.0_sbs.dat" in the migration/

MwsProgramFiles directory. When applying the composite template, the custom migration settings from the file are applied for the “mws_inst1”.

```
templates
  mws-server:
    fixes:
    products:
      MwsProgramFiles:
        mws_inst1:
          http.port: 8585
          db.url: ${db.url}
          db.username: ${db.username}
          db.password: ${db.password}
```

Provision

Each layer maps to one or more nodes (installations) through a provisioning map defined in the `provision` section. The nodes that a layer maps to are either existing nodes or nodes that will be created when applying the template. Based on the policy that you set for an environment type `innodes:nodeAlias:aliasMapping:policy:`, each node is referenced by:

- A global node alias, when `policy: HOST|EXISTING`
- An indexed node alias when `policy: INDEX`

For details about `aliasMapping:policy:` see ["Nodes" on page 88](#).

In the `provision` section, you can map each layer to a list of nodes or you can include a `default` section in which you specify a default mapping that applies to environment types or nodes that do not have an explicit definition. The `provision` section can also include references to lists of node aliases.

provision:envType:infrastructure

When `aliasMapping:policy: INDEX`, you must define the first layer in each `provisioning:envType` section as an *infrastructure* layer. The infrastructure layer does not have a reserved name, but it must come first in the `environment.type` section. This layer includes the `hosts` parameter that lists all Platform Manager nodes used by all layers defined in the template. For example:

```
provision:
  default:
    infra:
      hosts: [host1, host2, host3]
```

With the `INDEX` policy, each of the layers in `provision` (except the infrastructure layer) includes one the following parameters:

- `aliases`: A list of indexed node aliases. The indexed node alias is formed from the value of the `aliasMapping:policy:prefix` parameter in the `nodes` section and an index number that indicates the consecutive order of the host on the list in the `infrastructure:hosts` parameter. Use “_” as a separator in the indexed node alias.
- `indexes`: A list of index numbers that indicate the consecutive order of each host on the list in the `infrastructure:hosts` parameter.

The index numbers in `aliases:` and `indexes:` always start from 1.

Examples

- In the following example, the template uses the `INDEX` mapping policy. The "infra" layer is the infrastructure layer in the "default" and "envType1" sections and lists all hosts used for all layers defined in the template. By default "layer1" maps to the nodes with indexed aliases "default_node_1" and "default_node_2" and is installed on "host11" and "host12" in the `/opt/softwareag/sag_default_node` installation directory. In "envType1", "layer1" maps to the nodes with indexed aliases "my_node_3" and "my_node_4" and is installed on "host13" and "host14" in the `/opt/softwareag/sag_my_node` installation directory. The "esb" layer is installed on "host13" and "host14".

```
provision:
  default:
    infra:
      hosts: [host11,host12,host13,host14]
    layer1:
      aliases: [default_node_1,default_node_2]
    esb:
      indexes: [3,4]
  envType1:
    infra:
      hosts: [host11,host12,host13,host14]
    layer1:
      aliases: [my_node_3,my_node_4]
nodes:
  default:
    aliasMapping:
      policy: INDEX
      prefix: default_node
    default:
      bootstrapInfo:
        installDir: /opt/softwareag/sag_${node.alias.prefix}
  envType1:
    aliasMapping:
      policy: INDEX
      prefix: my_node
```

- In the following example, the template uses the `HOST` mapping policy. By default the "layer1" and "esb" layers map to the installation with node alias "node1". In "envType1", "layer1" maps to installations with aliases "node1" and "node2". The "esb" layer maps to installations with aliases "node2" and "node3". "envType2" maps layers to a list of user-defined node aliases.

```
provision:
  default:
    layer1: node1
    esb: node1
  envType1:
    layer1: [node1,node2]
    esb: [node2,node3]
  envType2:
    layer1: ${layer1.hosts}
    esb: ${is.hosts}
nodes:
  default:
    aliasMapping:
```

```
policy: HOST
```

Nodes

The `nodes` section defines the managed installations, also called nodes, to which the layers in the composite template are mapped.

You can define a list of nodes in a `nodes/nodeAlias` section, for each `environment.type` included in a composite template. All parameters that you specify for a node are optional. When you do not specify parameters for a node in a `nodeAlias` section:

- The node uses the properties specified in the `nodes/default/default` section and the node alias is also used as a host name for that node. This section specifies the properties of the default node definition.
- If the properties are not specified in the `nodes/default/default` section, the node uses the system default parameters.

Defining the local node in the `nodes` section is optional. Even when the local node is not defined in the `nodes` section, you can map the local node to layers in the `provision` section.

aliasMapping/policy

Important: This section is supported only with DSL version 1.1.

Based on the policy that you set in `nodes:default:aliasMapping:policy:` (for all environment types) or `nodes:envType:aliasMapping:policy:` (for a specific `environment.type`), each node is referenced by:

- A global node alias, when `policy: HOST|EXISTING`
 - `HOST` (default) maps each layer defined in the `provision/environment.type` sections to a list of node aliases or hostnames.
 - `EXISTING` uses the details of a Platform Manager node that already exists in Command Central (such as Platform Manager installed from the web user interface, the CLI, or another template) and ignores any parameters included in the `nodes` section of the template, except the node alias.
- An indexed node alias when `policy: INDEX`
 - `INDEX` uses a node alias generated from a custom prefix (specified in the `prefix:` parameter) and an index number, based on the consecutive order of the hosts in `provision:envType:infrastructure:hosts`.

When `policy: INDEX`, you must also include the `prefix:` parameter. The value of the `prefix:` parameter is used to generate an indexed node alias in `provision:envType:layer:aliases`. For example:

```
nodes:
  default:
    aliasMapping:
```

```
policy: INDEX
prefix: node
```

In the `nodes` section, you can also use reference parameters to refer to the node alias prefix and index. For example:

```
nodes:
  default:
    aliasMapping:
      policy: INDEX
      prefix: ${node.alias.prefix} # The prefix of the node alias.
    default:
      port: ${spm.port}
      bootstrapInfo:
        installDir: /opt/sag${node.alias.prefix}${node.index} # The indexed node alias
                                                              # (prefix plus index number)
```

See also the example of mapping layers to indexed node aliases in the ["Provision" on page 86](#) section.

credentials: \${spm.credentials.alias}

If Platform Manager is already bootstrapped on the target node and you use a custom password for the administrator user account, you must include the custom administrator user credentials in the `nodes` section of the composite template. In the following template snippet, the `credentials` parameter refers to the `spm.credentials.alias` parameter.

```
nodes:
  default:
    default:
      port: ${spm.port}
      secure: ${spm.secure}
      credentials: ${spm.credentials.alias}
```

You can specify the value of the `spm.credentials.alias` parameter in the `environments/default` section, a separate properties file, or as argument of the `apply` composite template command. The `spm.credentials.alias` parameter maps to the alias of the common credentials configuration instance you created for your custom administrator user, for example: `spm.credentials.alias: SECURE_ADMINISTRATOR`

where `SECURE_ADMINISTRATOR` is the alias of the custom `COMMON-CREDENTIALS-SECURE_ADMINISTRATOR` configuration instance. For information about creating common credentials configuration instances, see ["Configuration Types for Command Central and Platform Manager OSGI ENGINE" on page 475](#).

Example

In the following example, the installations with aliases "node1" and "node2" are installed only when creating an environment with type "envType1". The two nodes are installed on the local host with unique port numbers and installation directories.

The definition for "node1" and "node2" does not specify user credentials. The two nodes use the "DEFAULT_ADMINISTRATOR" user credentials specified in the `credentials` parameter in the `nodes/default/default` section. For information about the default common credentials configuration instances, see ["COMMON-CREDENTIALS Usage Notes" on page 477](#).

The `nodes/default/default` section does not provide a value for the HTTPS port and all nodes in the template use the system default value for the HTTPS port.

Platform Manager is installed on both nodes using the Command Central bootstrap installer specified in the `cc.installer` parameter in the `nodes/default/default/bootstrapInfo` section.

```
nodes:
  default:
    default:
      port: 8093
      secure: false
      credentials: DEFAULT_ADMINISTRATOR
      bootstrapInfo:
        cc.installer: ${}
        installDir: /opt/softwareag
        platform: lnxamd64

    entType1:
      node1:
        host: localhost
        port: 8192
        bootstrapInfo:
          installer: ${cc.installer}
          installDir: /opt/softwareag/n1

      node2:
        host: localhost
        port: 8292
        bootstrapInfo:
          installer: ${cc.installer}
          installDir: /opt/softwareag/n2
```

`nodes/default/default/bootstrapInfo`

In the `nodes/default/default` section, you can add an optional section with bootstrap information that enables you to bootstrap Platform Manager to a local or remote installation.

Usage Notes

Command Central attempts to connect to the Platform Manager host and port of the node, which Command Central wants to use, with the user credentials specified for the node. If Platform Manager is running and responds, Command Central uses the node. If Platform Manager does not respond, Command Central attempts to connect to the remote host using the SSH protocol and the user credentials defined in the `bootstrapInfo` section. If the connection is successful Command Central bootstraps and starts Platform Manager, and waits until Platform Manager becomes responsive and available to establish an HTTP/S connection.

Important: The remote host must have Java 8 installed and available for the SSH user. You can specify the location of Java on the remote host by adding a `javaPath` property in the `bootstrapInfo` section as follows:

```
javaPath: /path/to/java
```

You can specify the following parameters in the `nodes/default/default/bootstrapInfo` section:

```
bootstrapInfo:
  installer: ${bootstrap.installer}      #The file name of the Command Central
                                        #bootstrap installer to use
                                        #for bootstrapping Platform Manager.
  installDir: ${install.dir}            #The installation directory
                                        #of Platform Manager.
  credentials: ${credentials.alias}     #For remote bootstrap: The SSH credentials
                                        #to use to connect to the remote hosts.
  version: "${release}"                 #The release version of
                                        #the Platform Manager node.
                                        #For backward compatibility when "cc.installer"
                                        #is not available.
```

Note: In the `nodes/default/default/bootstrapInfo` section, the following parameters are deprecated beginning with release 10.1:

- The "repo.spm" and "platform" parameters are replaced by the "installer" parameter.
- The "username" and "password" parameters are replaced by the "credentials" parameter.

Example

The following template snippet shows how to define a composite template that you can use to bootstrap Platform Manager to a local node. The alias of the new Platform Manager installation, "dev8192", is specified in the "spm-alias" parameter in the `environments/default` subsection. The "cc.installer" parameter is the Command Central bootstrap installer to use to install Platform Manager. The "install.dir" parameter is the path to the Platform Manager target installation directory. The "cc.installer" and "install.dir" parameters are required and their values depend on your operating system. You must provide values for those parameters at the time of applying the template. The "install.dir" value should point to a directory that does not exist on the target node and for which the Command Central user account that you use to bootstrap Platform Manager has write access. The "spm.port" default value is unique and is not used by any other local process, including Command Central and Platform Manager.

The parameters in the `nodes/default/default/bootstrapInfo` section refer to the values for Command Central bootstrap installer and Platform Manager installation directory specified in the `environments/default` section. The `nodes` section also defines an "spm.alias" node with host "localhost". The `layers` section defines a single Platform Manager "management" layer that maps to the "spm.alias" node in the `provision` section.

```
alias:bootstrap
description: Bootstrapping local nodes

environments:
  default:
    cc.installer: ${}
    install.dir: ${}
    spm.port: 8192
    spm.alias: dev${spm.port}
```

```

nodes:
  default:
    default:
      port: ${spm.port}
      secure: false
      credentials: DEFAULT_ADMINISTRATOR
      bootstrapInfo:
        installer: ${cc.installer}
        installDir: ${install.dir}

    ${spm.alias}:
      host: localhost

layers:
  management:
    description: management layer of SPM
    templates: spm-tuneup
templates:
  spm-tuneup:
    products:
      SPM:

provision:
  default:
    management: ${spm.alias}

```

Repositories

In the `repositories` section, you can define master, image, and mirror repositories for products, fixes, and assets. You can reference repositories that are already defined in the template.

When you define an image repository, the repository definition must include the `location` parameter that specifies the path to the image file that contains the products or fixes:

```

repositories:
  product:
    My-10.1-linux:
      location: path/to/product/image/file.zip
  fix:
    My-10.1-fixes:
      location: path/to/fix/image/file.zip

```

When you define an asset repository, you must include the `type` and `location` parameters. The `type` parameter indicates the type of the repository (flatfile or git) and the `location` parameter specifies the path to the flatfile repository or the URL of the git repository. For example, to define an asset repository located in git:

```

repositories:
  asset:
    assets-repo:
      type: git
      description: Test assets
      location: ssh://git@github.com/sag-test/test.git
      credentials: NONE

```

In this example, `credentials: NONE` indicates that the git repository is public and does not require authentication to clone the repository. For information about the default

common credentials configuration instances, see "[COMMON-CREDENTIALS Usage Notes](#)" on page 477.

To specify the user credentials required to connect to a repository, you can create an instance of the COMMON-CREDENTIALS type (with the authentication details for the repository), using the Command Central web user interface or the `sagcc create configuration data` command. Add the `credentials` parameter and refer to the alias of the COMMON-CREDENTIALS configuration instance you created for the repository as follows:

```
repositories:
  product:
    webMethods-${version}:
      credentials: ${repo.product.credentials.alias}
  fix:
    Empower:
      credentials: ${repo.fix.credentials.alias}
  asset:
    assets-repo:
      credentials: ${repo.asset.credentials.alias}
```

You can specify the value for the `repo.product.credentials.alias`, `repo.product.credentials.alias`, and `repo.asset.credentials.alias` parameters in the `environments/default` section, a separate properties file, or as argument of the `apply composite template` command. The parameters map to the alias of a common credentials configuration instance, for example: `repo.product.credentials.alias: EMPOWER-CRED`, where `EMPOWER_CRED` is the alias of the custom `COMMON-CREDENTIALS-empower_cred` configuration instance that defines the credentials for the Empower product and fix repositories. For information about creating common credentials configuration instances, see "[Configuration Types for Command Central and Platform Manager OSGI ENGINE](#)" on page 475.

You can also specify the alias directly as the value of the `credentials` parameter, for example:

```
repositories:
  product:
    webMethods-10.1:
      credentials: EMPOWER-CRED
  fix:
    Empower:
      credentials: EMPOWER-CRED
```

For an example of a repository definition, see the `repos` sample template, located here: <https://github.com/SoftwareAG/sagdevops-9.x-template-samples>.

Actions

In the `actions` section, you can define shell actions that the composite template executes at the time of applying the template in both provisioning and migration mode. For example, if you want to ensure that you have the system resources required for a provisioning operation, you can define a shell action in the composite template definition that will check the system resources before starting the provisioning operation.

The actions defined in the template are executed as remote actions on the target hosts, using the details of the SSH connection defined for the target host in the `bootstrapInfo` section under `nodes`. The actions are executed on the Command Central server host as local actions only when:

- The actions are defined in the `environments` section.
- The actions are executed on the local node.

Each action is defined either inline in the composite template, or in an external file located on the Command Central server, the target node, or in the composite template archive. Based on the section in which you include an `actions` section, Command Central will execute the actions as follows:

<u>The actions in this section...</u>	<u>are executed...</u>
<code>environment/default/actions</code>	on the local node for all types of environments.
<code>layers/default/actions</code>	on the remote nodes for all layers, except for the layers that map to the local node.
<code>templates/default/actions</code>	on the remote nodes for all templates, except for the templates applied on the local node.
<code>nodes/default/actions</code>	on all nodes. If the action is executed for the local node, it is a local action.
<code>environments/<i>environment.type</i> / actions</code>	on the local node, only for this environment type.
<code>layers/<i>layerAlias</i> / actions</code>	when processing the layer with the specified alias. If the layer maps to the local node, the actions are executed as local actions.
<code>templates/<i>templateAlias</i> / actions</code>	when processing the inline template with the specified alias. If the template is applied on the local node, the actions are executed as local actions.
<code>nodes/<i>nodeAlias</i> / actions</code>	on the node with the specified alias.

Action Parameters

The actions section includes the following parameters:

Parameter	Description
<code>actionName:</code>	Required. The name of the action. If the action is executed on Windows, the name of the action ends with ".bat", for example: <code>actionName.bat</code> :
<code>description:</code>	Optional. States the goal of the action.
<code>phase: {pre post}</code>	<p>Specifies whether to execute the actions before or after the operation defined in the composite template section. For example, in the <code>nodes/default/actions</code> section, if <code>phase=pre</code>, the actions are executed before bootstrapping Platform Manager. If <code>phase=post</code>, the actions are executed after bootstrapping Platform Manager.</p> <p>The actions defined in the default sub-sections of top level sections are executed as follows:</p> <ul style="list-style-type: none"> ■ Default actions with the <code>phase=pre</code> parameters are executed before regular <code>pre</code> actions. ■ Default actions with the <code>phase=post</code> parameters are executed after regular <code>post</code> actions.
<code>failOnError: {true false}</code>	<p>Indicates whether applying the composite template will fail if the action fails with an error. Valid values:</p> <ul style="list-style-type: none"> ■ <code>true</code> (default) - The composite template apply operation fails. ■ <code>false</code> - The composite template apply operation ignores the error and continues processing the composite template.
<code>script:</code>	<p>Required when you do not specify the <code>file</code> property. Includes script commands, defined inline.</p> <p>When the shell scripts are included inline, the shell variables using the notation <code>\${var}</code> are evaluated against the properties defined in the composite template. The shell variables using the notation <code>\$var</code> are not evaluated against the properties defined in the composite template and the script is executed as it stands.</p>

Parameter	Description
	If you include both <code>script</code> and <code>file</code> properties, Command Central uses the <code>script</code> property and ignores the <code>file</code> property.
<code>file:</code>	Required when you do not specify the <code>script</code> property. Specifies either the absolute, or the relative location path to an external script file. If you specify a relative location path to the composite template archive, the file is taken from the template archive. For example, <code>file: scripts/main.sh</code>
<code>target: POSIX WINDOWS</code>	Optional. Indicates on which operating systems to execute the action. Valid values: <ul style="list-style-type: none"> ■ <code>POSIX</code> (default) - the action is executed on POSIX-compliant operating systems. ■ <code>WINDOWS</code> - the action is executed only on Windows operating systems. This value is not supported for remote actions.

Usage Notes

- The script code in the shell action can use any interpreter available on the system, such as Perl, Python, or Ruby.
- When you define actions under the `templates` section and you apply the template with `environment.mode=provision`, even if `failOnError: true`, the template does not fail if the action fails.
- When configuring the `target` property for an action, you must consider the following:
 - For a local action, set `target: WINDOWS` if the operating system of the local node is Windows and you want to use a Windows script. If the local node has a Unix operating system and you want to use a Unix script, set `target: POSIX`.
 - For a remote action executed on a target host with a Windows operating system, you must set `target: POSIX`, but ensure that the target host is configured for an SSH connection with a third-party tool, for example Cygwin. For information about how to install Cygwin, go to <https://cygwin.com/install.html>.

Example

In the following example, the actions section defines a “cleanUpEnvDefault” action that the composite template will execute after the provisioning operation for all environment

types. The composite template application will fail if the action fails with an error. The script command is specified inline in the “script” property.

The “\${temporary.data}” variable in the “rm -r” argument of the inline script is evaluated against the properties defined in the composite template. If the composite template definition includes a property with name “temporary.data”, the script will use the value of that property. If the template does not include a “temporary.data” property, the script is executed as it stands.

```
environments:
  default:
    actions:
      cleanUpEnvDefault:
        description: "Default clean up of the environment after provisioning"
        phase:post
        failOnError: true
        script: |
          #!/bin/sh
          rm -r${temporary.data}
```

Format of COMMON-* Configuration Types in the Template Definition

The following table lists the supported formats for the parameters you specify for each COMMON-* configuration type of a product instance in the template definition (template.yaml file):

Configuration type	Format in the template definition
COMMON-CLUSTER	YAML
COMMON-COMPONENT-ENDPOINTS	YAML
COMMON-CREDENTIALS	YAML
COMMON-DBFUNCTION	YAML
COMMON-JAAS	text/plain
COMMON-JAAS-REALMS	text/plain
COMMON-JAVASYSPROPS	text/plain
COMMON-JDBC	YAML

Configuration type	Format in the template definition
COMMON-JMS	YAML
COMMON-JNDI	YAML
COMMON-JSW	text/plain
COMMON-JVM-OPTIONS	YAML
COMMON-KEYSTORES	YAML
COMMON-LDAP	YAML
COMMON-LICENSE	text/plain
COMMON-LICLOC	YAML
COMMON-LOCAL-GROUPS	YAML
COMMON-LOCAL-USERS	YAML
COMMON-LOG	YAML
COMMON-LOG4J	YAML
COMMON-LOGGERS	YAML
COMMON-MEMORY	YAML
COMMON-PORTS	YAML
COMMON-PROXY	YAML
COMMON-SMTP	YAML
COMMON-SYSPROPS	text/plain
COMMON-TRUSTSTORES	YAML

Configuration type	Format in the template definition
COMMON-VARS	YAML
COMMON-WMMESSAGING	YAML

Important: When using multi-line plain text, you must insert a white space before the pipe symbol (|).

Examples

- Using text/plain format:

```
COMMON-SYSPROPS:
  COMMON-SYSPROPS-Global_Values: |
    HTTPCookieSize=20
  COMMON-SYSPROPS-Connection_Config: |
    MaxBufferSize=31457280
    AllowBufferReuse=false
```

- Using YAML format:

```
COMMON-MEMORY:
  COMMON-MEMORY:
    InitSize: ${um.memory.init}
    MaxSize:  ${um.memory.max}
    ExtendedProperties:
    Property:
      -
        "@name": "-XX:MaxDirectMemorySize"
        $:  ${um.memory.direct}
```

Summary of Sample Composite Templates

You can find sample composite templates here: <https://github.com/SoftwareAG/sagdevops-9.x-template-samples>. You can use the sample composite templates to set up basic or typical product environments with Software AG products of release 9.x.

Note: Software AG recommends that you use the sample micro templates located here: <https://github.com/SoftwareAG/sagdevops-templates> to set up environments with Software AG products of release 10.1 and higher.

The following table lists the sample composite templates, as well as their environment types, required parameters, and description:

Template alias	Environment Types	Required Parameters	Description
bpms	<ul style="list-style-type: none"> ■ dev ■ server ■ cluster 	<ul style="list-style-type: none"> ■ database connection ■ host names ■ database server administrator user credentials ■ license key aliases for Universal Messaging, Terracotta Server Array, Integration Server, Rules Engine 	<p>Provisions a typical BPMS environment, configured with the following layers:</p> <ul style="list-style-type: none"> ■ messaging (Universal Messaging) ■ cache (Terracotta Server Array) ■ bpm (Integration Server/Process Engine/Rules Engine/Monitor) ■ presentation (My webMethods Server/Task Engine/Rules Engine/Monitor) <p>Use this template to set up a general purpose BPMS environment. The template requires connection to an external relational database. The database user and schema are created automatically.</p>
dbc	<ul style="list-style-type: none"> ■ default ■ dev 	<ul style="list-style-type: none"> ■ database connection ■ database server administrator user credentials ■ database components ■ database products 	<p>Installs the Database Component Configurator (DBC) on a local or remote installation and uses DBC to create a new database on a supported database server.</p> <p>You must specify the database administrator user credentials to be able to create the new database storage and user.</p> <p>Before using the template, read the readme file located in the dbc template directory.</p>

Template alias	Environment Types	Required Parameters	Description
is-layer	<ul style="list-style-type: none"> ■ dev ■ server ■ layer ■ cluster 	<ul style="list-style-type: none"> ■ database connection ■ host names ■ Terracotta Server Array URL ■ database server administrator user credentials ■ license key alias for Integration Server ■ license key alias for Terracotta 	<p>Provisions a standalone local or remote Integration Server, or an Integration Server cluster with multiple hosts.</p> <p>Use this template to setup an integration layer based on Integration Server.</p> <p>When provisioning an Integration Server cluster, you must provide an external relational database and an external Terracotta layer.</p> <p>Before using the template, read the readme file located in the is-layer template directory.</p>
mws-layer	<ul style="list-style-type: none"> ■ dev ■ server ■ cluster 	<ul style="list-style-type: none"> ■ database connection ■ host names ■ database server administrator user credentials 	<p>Provisions a standalone local or remote My webMethods Server, or a My webMethods Server cluster with multiple hosts.</p> <p>With this template, you can setup a presentation layer using My webMethods Server.</p> <p>When provisioning a My webMethods Server cluster, you must provide an external relational database.</p>
repos	default	Empower user name and password	<p>Contains the public master product and fix repositories of release 9.8 and higher, except the repositories of the latest release.</p> <p>These repositories are required when</p>

Template alias	Environment Types	Required Parameters	Description
			<p>bootstrapping installations or executing template-provisioning operations.</p> <p>Apply this template first on the master Command Central installation that has Internet access to the Empower Product Support Website.</p>
spm-layer	<ul style="list-style-type: none"> ■ default ■ dev ■ server ■ layer 	<ul style="list-style-type: none"> ■ SSH connection ■ host names 	<p>Bootstraps a single local or remote Platform Manager, or a number of remote Platform Managers on UNIX hosts via an SSH connection.</p> <p>Use this template to setup a base management layer on which to install core products, using composite templates.</p>
tc-layer	<ul style="list-style-type: none"> ■ dev ■ server ■ cluster 	<ul style="list-style-type: none"> ■ host names ■ license key alias for Terracotta Server 	<p>Provisions a standalone local or remote Terracotta Server or a two-server master/slave cluster on remote machines.</p>
um-layer	<ul style="list-style-type: none"> ■ dev ■ server ■ cluster 	<ul style="list-style-type: none"> ■ host names ■ license key alias for Universal Messaging 	<p>Provisions a standalone Universal Messaging realm instance or a two-instance cluster on local or remote machines.</p> <p>Use this template to set up a messaging layer using Universal Messaging.</p>

Using the is-layer Sample Template

You can use the is-layer sample composite template as a standalone template to provision a basic environment with a core Integration Server instance, or as part of a complex composite template, for example to create a Business Process Management environment, based on Integration Server. The sample is-layer composite template is located here: <https://github.com/SoftwareAG/sagdevops-9.x-template-samples>.

Before Using the is-layer Sample Template

1. Install Command Central as described in the getting started with Command Central pages in this help.
2. In the Repositories view in the Command Central web user interface, register product and fix repositories to use with the is-layer default template. You can also use the command line interface as follows:

- Use the `sagcc add repository products` and `sagcc add repository fixes` commands.
- Apply the `repos` default composite template with this command:

```
sagcc exec templates composite apply repos "empower.username=email "
"empower.password=password"
```

where *email* and *password* are your log-on user credentials for the Empower web site.

- To verify that the product and fix repositories you require are added in Command Central, use the `sagcc list repository` command.

3. Add the license key files of Integration Server and Terracotta Server using the "`sagcc add license-tools keys`" on page 320 command. For example, to add a license key for Integration Server 9.12 and assign to that key the alias "Integration_Server912-w64":

```
sagcc add license-tools keys Integration_Server912-w64 -i licenseKey.xml
```

To add a license key for Terracotta:

```
sagcc add license-tools keys Terracotta -i Terracotta.key
```

The license key aliases that you assign to Integration Server and Terracotta Server are used in the template definition to identify the license key files that the Terracotta Server product instance requires.

4. Ensure that the UNIX or Windows operating system on the target machine has a Secure Shell (SSH) server running, for example OpenSSH, and the system is configured for remote access with the user account for the Software AG products.

To set up OpenSSH on Windows, you can use a third-party tool, for example Cygwin. For information about how to install Cygwin, go to <https://cygwin.com/install.html>. When installing Cygwin, follow the instructions about setting up OpenSSH located here <http://www.noah.org/ssh/cygwin-sshd.html>.

If the target machine does not have an SSH server, you must install only Platform Manager on the remote machine as described in the getting started with Command Central pages in this help.

5. Configure the is-layer default template as described in "[Configuring the is-layer Template](#)" on page 104.

Configuring the is-layer Template

To configure the is-layer default template, you should:

1. Read the readme.md file, included in the is-layer zip archive, to understand the requirements and typical use cases for the template.
2. Read the "[Understanding the is-layer Template Definition](#)" on page 104 to understand the design of the is-layer template definition and how Command Central processes each section of the template.
3. Configure the is-layer template using an external properties files:
 - a. Copy the sample.properties file to a location that you can later add under a version control system.
 - b. Rename the sample.properties file.
 - c. Open the renamed file in a text editor and specify values for the required properties.

When you configure the environment properties in a separate file, you can apply the same template definition with different environment properties to create different types of environments.

Important: Do not edit or modify the template.yaml file of the is-layer default template in any way. If you want to use the template.yaml file of the is-layer template to create a custom template definition, for example add user-defined layers or inline templates, you must create a custom composite template as described in "[Creating a Custom Composite Template](#)" on page 120.

Understanding the is-layer Template Definition

Environments

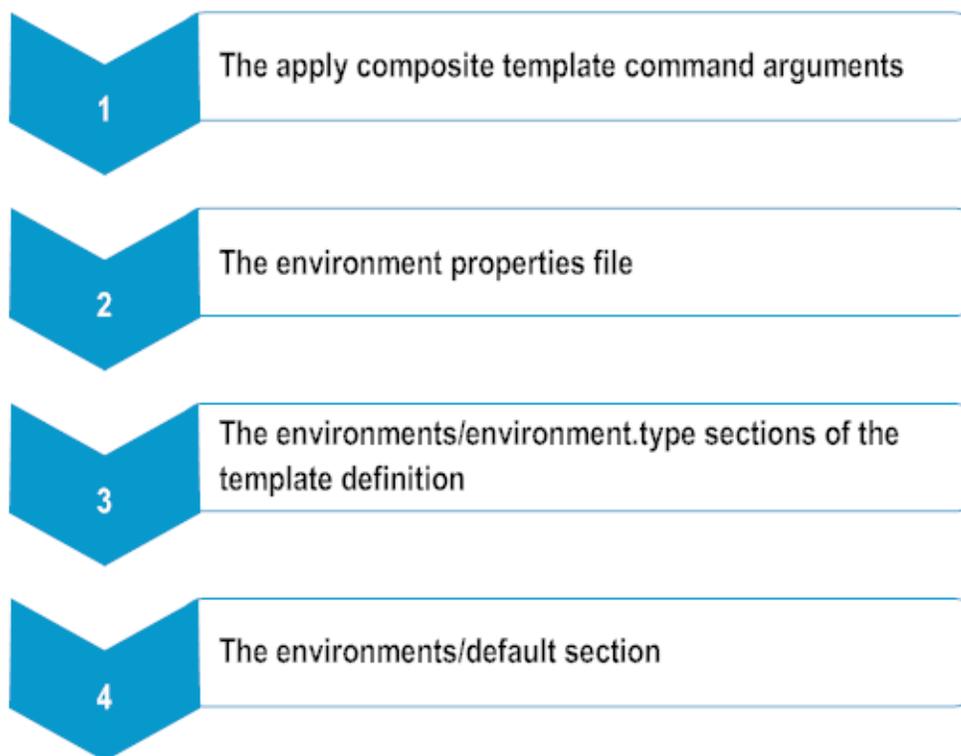
In this section you specify what type of environments you want to create on the target hosts. The is-layer template defines the following types of environments:

- `dev` - a local development environment
- `server` - an environment with a single Integration Server instance
- `layer` - an environment with multiple Integration Server instances, not configured in a cluster

- `cluster` - an environment with Integration Server instances configured in a cluster, with a Terracotta Server

When you define environment properties that apply for all types of environments, you include them under the `environments/default` section. If an environment property applies only for a particular type of environment, you include it under the respective environment section. For example, the `spm.alias` property applies only for the `dev` environment, because it is in the `environments/dev` section. Conversely, the `environments/server` section does not have any parameters and the server environment uses the environment properties defined in the `environments/default` section.

In general, when the `is-layer` template is used to install a new environment, Command Central determines which values to use for the environment properties in the following order:



For example, the `dev` environment uses the value of `spm.port` from the `environments/dev` section, instead of the `environments/default` section.

Some of the properties in the `environments` section have a default value, which you can specify in one place and then refer to that property/value pair from other parameters defined in the template. For example:

```
repo.spm: ${repo.product}
```

indicates that `repo.spm` refers to the `repo.product` property, which in the `sample.properties` file has the default value: `repo.product=webMethods-`

`${version}_GA`. The `version` property, included in `webMethods-${version}_GA` also has a default value: `version=9.12`. When you apply the template without command arguments, Command Central will use the default value of `repo.product` from the properties file to determine from which product repository to install products.

Note that if the value of a parameter is specified as follows:

```
is.host: ${}
```

you must provide a value for this parameter in the template definition, a properties file, or a command argument. Otherwise, the template fails to apply successfully.

The `environments/default` section groups the environment properties, based on which information for the new environment they define:

- The product and fix repositories from which to install Platform Manager and the products, and apply the required fixes.
- The system configuration settings of the remote machine, required to bootstrap Platform Manager.
- The configuration settings required to create an Integration Server instance.
- The database configuration details, required to create the database schema and storage for Integration Server, and to enable Integration Server to connect to the database server.

Layers

In the `layers/default` section, the `productRepo` and `fixRepo` refer to the repository parameters, defined in the `environments/default` section and identify the product and fix repositories from which to install product and fixes for all layers, defined in the section. Each layer has the following purpose and properties:

- the `spm` layer is the first layer that Command Central applies, because this layer ensures that Platform Manager is updated to the required fix level and the `spm-tuneup` inline template is applied to install the product plug-ins. The `templates` parameter refers to the `spm.configure` parameter defined in the remote system configuration section of `environments/default`, which in turn points to the `spm-tuneup` inline template. The `spm.fixes` parameter identifies the repository from which to install the Platform Manager fixes.
- The `dbc` layer is used to create database storage and schemas for Integration Server on the target node. The `templates` parameter refers to the `dbc.configure` parameter defined in the `environments/default` section, which in turn points to the `dbc` inline template.

In the `databases` section you must define the details required for creating the database storage and schemas. The properties in the `dbc-components` and `dbc-products` sections point by reference to parameters defined in the `environments/default` section. Some of the parameters have default values to ensure that the database storage and schema are created and updated correctly, for example:

```
environments:
  default:
    db.components: [STR]
```

```
db.products:          [IS]
db.component.version: latest
db.product.version:  latest
```

Other parameters do not have defaults and you must specify user-defined values:

```
environments:
  default:
    db.type:          ${} # REQUIRED: Database type: oracle,sqlserver,db2
    db.host:          ${} # REQUIRED: Database server hostname
    db.port:          ${} # REQUIRED: Database server port
    db.admin.username: ${} # REQUIRED: for db storage only
    db.admin.password: ${} # REQUIRED: for db storage only
    db.tablespace.dir: ${} # for Oracle/DB2
```

When applying the `dbc` layer on a target node of release 9.9 or lower, the `dbc.alias` parameter to which you map the `dbc` layer in the provisioning section must point to the `local` node. Beginning with release 9.10, the `dbc.alias` can point to a local or remote Platform Manager node, for example:

```
environments:
  default:
    dbc.alias: remotehost
```

Important: The host of the node on which the `dbc` layer is applied must have access to the database server host to be able to execute the database operations.

- The `is` layer is used to install Integration Server, and then create and configure the Integration Server instances. The `templates` parameter refers to the `is.configure` parameter defined in the Integration Server instance configuration section of `environments/default`, which in turn points to the `is-server` inline template.

Templates

This section includes the following inline templates:

`spm-tuneup`

The `products` section lists the IDs of Update Manager, Platform Manager, and the Integration Server plug-in for Platform Manager to ensure that they are installed on the Platform Manager node. After installing those components, Command Central applies the Platform Manager fixes, defined in the `fixes` property. The fix installation operation restarts Platform Manager, which enables the Integration Server plug-in.

`dbc`

The `products` section lists the ID of the Database Component Configurator (DBC) to ensure that this component is installed on the Platform Manager node. The DBC component is required for the execution of the database actions on the target nodes.

`is-server`

Command Central applies the `is-server` inline template in the following order:

1. Retrieves the Integration Server license file that matches the Integration Server key alias, defined in the `licenses` section, and copies the file to the "IntegrationServer/config/licenseKey.xml" location on the target node. Command Central performs the same action for the Terracotta Server: retrieves the

- Terracotta license file that matches the license key alias for Terracotta and copies it to the "common/conf/terracotta-license.key" target location.
- Identifies the Integration Server product ID from the `products` section and installs Integration Server.
 - Applies the Integration Server fixes, defined in the `fixes` section. This section refers to the `is.fixes` parameter, defined in the remote system configuration part of the environments/default section.
 - Creates an Integration Server instance with the "default" alias specified in the `${is.instance.name}` parameter. The Integration Server instance is created with the ports and database connection details, specified in the properties under the `${is.instance.name}` section.
 - Applies the configuration properties defined for the Integration Server instance in the `configuration` section. The configuration properties for the Integration Server OSGi profile and for the Integration Server instance are in separate sections, and the properties for each configuration type have their own subsection. List the properties for each configuration type as parameters in YAML format or as items of a list. To introduce a list of items, prefix the line with a dash, for example:

```
configuration:
  integrationServer-${instance.name}:
    COMMON-LOCAL-USERS:
      COMMON-LOCAL-USERS-Administrator:
        "@id": Administrator
        Enabled: true
        Password: ${is.password}
        ExtendedProperties:
          Property:
            -
              "@name": "AllowDigestAuthentication"
              $: false
```

Provision

Command Central uses the map in this section to determine which layers to apply on which target nodes for each environment type. The layers are mapped to the node aliases. When creating the new environment, Command Central applies one layer on all target nodes to which the layer maps before it starts applying the next layer. For example:

```
dev:
  spm: ["${spm.alias}"]
  dbc: ["${dbc.alias}"]
  is: ["${spm.alias}"]
```

In the example, when creating a dev environment, Command Central applies the `spm` layer on the host of the Platform Manager installation with alias "spm.alias". After applying the `spm` layer, Command Central applies the `dbc` layer on the host specified for "dbc.alias", and after the `dbc` layer is done, the "is" layer gets applied on the host of "spm.alias".

You also have the option to define a list of hosts on which to apply a layer as a parameter and refer to that list, for example:

```
layer:
  spm: ${is.hosts}
  dbc: ["${dbc.alias}"]
  is:  ${is.hosts}
```

In the example, when installing a layer environment, the `spm` and `is` layers are applied on the list of hosts defined for the `is.hosts` parameter in the properties file.

The nodes in the list are applied in parallel, based on the thread pool size. For example, if the thread pool size is set to 10 and you have 100 nodes in the list, the nodes are processed 10 at a time in parallel, in the order in which they are listed.

Nodes

Command Central uses the information from this section to determine how to install Platform Manager on the remote machine. This section includes two sections:

- `default` - properties used to install Platform Manager for all environment types
- `dev` - properties used for the `dev` environment type. When bootstrapping Platform Manager for the `dev` environment, Command Central will use the `spm.alias` as the alias and `is.host` as the host of the new Platform Manager installation. The remaining bootstrap details are taken from the `nodes/default/default/bootstrapInfo` section.

When Platform Manager is installed and running on the remote machine, Command Central skips the `bootstrapInfo` section, which means that you must only configure the `port`, `host`, and `secure` properties in the `nodes/default/default` section. Note that by default, `spm.secure` is set to `false`. To enable HTTPS, set `spm.secure: true`.

When Platform Manager is not installed on the remote machine, Command Central uses the properties in the `nodes/default/default/bootstrapInfo` section to determine how to install Platform Manager. The `installDir` property specifies the installation directory of Platform Manager on the target machine and `repoName` identifies from which product repository to install Platform Manager. The `platform` and `port` properties refer to the `os.platform` and `os.port` parameters, which identify the operating system and the SSH port of the remote machine.

The `distribution` property specifies how to install Platform Manager. Software AG recommends that you use:

- `DEF` for environments of release 9.10 and higher. Installs Platform Manager, Update Manager, and CLI.
- `ALL` for environments of release 9.8 and 9.9. Installs Platform Manager, Update Manager, CLI, and all product plug-ins.

The `credentials` section includes details about the SSH user account on the remote machine that owns the Software AG installation directory files and can execute Command Central processes and commands. All properties under `credentials` refer to configuration parameters of the remote operating system, defined in the `environments/defaults` section. The default value of the `os.auth.method` parameter is

the `PASSWORD` authentication method for the SSH connection to the remote machine. If you set `os.auth.method: CERTIFICATE`, you must also specify the location of the RSA private key file as the value of the `os.auth.key` parameter. For example: `os.auth.key: ${user.home}/.ssh/id_rsa`

Important: The remote host must have Java 8 installed and available for the SSH user. You can specify the location of Java on the remote host by adding a `javaPath` property in the `bootstrapInfo` section as follows:

```
javaPath: /path/to/java
```

Inside the Sample Properties File

The `sample.properties` file, included with the `is-layer` template, contains sample properties that you can configure and use to install Integration Server with an Oracle database. By default, the sample properties are defined for a local development environment. When you want to install a different type of Integration Server environment, you comment out all lines with properties for the `environment.type` sections that you do not want to use, and uncomment the lines of the `environment.type` section that you want to use. For example, if you want to install an Integration Server cluster, change the environment type sections as follow:

```
#####
# dev/server/layer/cluster
# environment.type=dev
#####
# environment.type=server
# is.host=ccdemowin2012
# os.platform=w64
os.username=vmtest
os.password=vmtest
#####
# environment.type=layer
is.hosts=[bgcctbp12,bgcctbp13]
os.platform=lnxamd64
#####
environment.type=cluster
is.cluster.enabled=true
is.tsa.url=bgcctbp12:${port.range}10
tc.license.key.alias=Terracotta
```

Applying the is-layer Sample Template

To apply the `is-layer` template, using only the parameter values defined in the template definition or an external properties file, run the `apply` composite templates command with the `input file` option, for example:

```
sagcc exec templates composite apply is-layer -i env.properties
```

where `env.properties` is your custom properties file.

You can also provision just one or more of the environments defined in the template by adding command arguments, or using as many different `.properties` files as your target environments.

Examples

- To install a dev environment in the new installation directory "C:\sag912\mydevenv", located on the local machine, add the `environment.type=dev` argument:

```
sagcc exec templates composite apply is-layer environment.type=dev
repo.product=webMethods-9.12 repo.fix=Empower
is.license.key.alias=Integration_Server912 tc.license.key.alias=Terracotta
install.dir=C:\sag912\mydevenv
```

Command Central uses the product and fix repositories, and the Integration Server license key specified in the command arguments.

- To install an environment with a single Integration Server that uses an Oracle database on the "rubicon2" target node, add the `environment.type=server` and `is.host=rubicon2` arguments:

```
sagcc exec templates composite apply is-layer -i sample.properties
environment.type=server is.host=rubicon2
```

Command Central uses the connection details of the Oracle database from the `sample.properties` file.

- To install an environment with several instances of Integration Server, which are not configured in a cluster, on two Linux target nodes, add the `environment.type=layer` and `is.hosts=[rubicon1,rubicon2]` arguments:

```
sagcc exec templates composite apply is-layer -i sample.properties
environment.type=layer is.hosts=[rubicon1,rubicon2] os.platform=lnxamd64
```

- To install an environment with a cluster of Integration Server instances that uses a Terracotta Server, add the `environment.type=cluster` and `is.hosts=[rubicon1,rubicon2]` arguments:

```
sagcc exec templates composite apply is-layer -i sample.properties
environment.type=cluster is.hosts=[rubicon1,rubicon2] os.platform=lnxamd64
is.tsa.url=rubicon3:9910 tc.license.key.alias=Terracotta
```

In the example, the `is.tsa.url` argument specifies the URL that Integration Server must use to connect to the Terracotta Server and the `tc.license.key.alias` specifies the license key to use for the Terracotta Server.

If applying the template fails, see ["Correcting a Failed Composite Template Apply Operation" on page 117](#).

How Command Central Processes a Composite Template

Command Central applies a composite template based on the processing mode specified in the `environment.mode= {provision | migration}` argument of the [sagcc exec templates composite apply](#) command. When you execute the apply command without including the `environment.mode` argument, the default mode is `provision`. During the template application process, some steps are executed only in migration mode.

Command Central performs the following tasks during the application of a composite template.

Step 1

Validates the composite template, as described in the ["Validating a Composite Template" on page 114](#) topic.

Step 2

Processes the layers in the source environment. This step applies *only* when the `environment.mode` argument of the apply composite template command is set to `environment.mode=migration`. Command Central prepares the source environment for migration.

The layers are processed one at a time, starting from the first layer defined in the template and going down the list of layers. For example if the `layers` section defines `um`, `is`, and `mws` layers, the processing starts with the `um` layer. Command Central goes through the following processing steps for each layer in turn:

1. Identifies the target set of nodes from the `provision/environment.type /layerAlias` section of the template definition.
2. Identifies the source nodes from the `migration/source/environment.type` section. If the source nodes are not defined in this section, by default Command Central uses the same set of nodes as both source and target nodes.

Note: When the template included a `migration/source/default` section, but does not have `migration/source/environment.type` sections, if `environment.type` sections exist for the target nodes, the template will again use the properties defined for the target nodes, instead of the `migration/source/default` properties.

3. Identifies the host, port, and installation directory of the source nodes from the `migration/nodes/environment.type /node.alias` section. If no properties are specified in the `node.alias` section, Command Central uses the values for host, port, and installation directory specified for the target nodes in the top level `nodes` section.
4. Prepares each source node for migration. By default, the following processing steps are executed in parallel for all source nodes:

Tip: You can customize the migration settings in the `migration/options` section.

Processing step	Description
1	Create a template with products, fixes, configuration, and files from each source node.
2	PAUSE all run-time components:

Processing step	Description
	<ul style="list-style-type: none"> ■ If the PAUSE operation is not supported, the run-time component returns an error, which is ignored. ■ If the PAUSE operation is successful, Command Central waits for the run-time component to report PAUSED status.
3	<p>Shut down all run-time components, including Platform Manager.</p> <p>Note that you can set the <code>shutdown</code> option to <code>false</code> only when migrating to a different host on a Unix operating system. For all other types of migration, this option is set to <code>true</code>.</p>
4	<p>Rename the source installation directory to <code>migration_source_installDir</code> when migrating on the same host and installation directory.</p>

At the end of the source environment processing step, each layer has a template.

Step 3

Bootstraps Platform Manager on the target nodes.

The bootstrap operation is executed for all layers and runs in parallel on all target nodes. The target set of nodes is identified from the `provisioning/environment.type` sections.

At the end of the bootstrap processing step, Platform Manager is installed on all target nodes.

Step 4

Provisions or migrates all layers to the target nodes.

The layers are processed one at a time, starting from the first layer defined in the template and going down the list of layers. For example if the `layers` section defines `is`, `mws`, and `um` layers, the processing starts with the `is` layer. Command Central goes through the following processing sub-steps for each layer in turn:

1. Applies each layer on the target nodes to which the layer maps in the provision section. The nodes get processed one node at a time. Command Central maps the source nodes to the target nodes using the node alias.
2. Applies the inline templates listed for each layer.

At the end of the layers processing step, each layer is applied on the specified target nodes.

Step 5

Installs products and fixes from the inline templates defined in the composite template as follows:

1. Install products, including the Platform Manager product plug-ins.
2. Install all current fixes for the installed products and plug-ins.

At the end of the product and fixes provisioning step, the products and fixes included in the templates are installed on the target nodes.

Step 6

Creates, updates, or migrates the product instances from the templates defined in the composite template.

When Command Central migrates the environment, each product instance is migrated using the migration utility of the product as follows:

1. The `instance_name` parameter from the template is passed to the product migration API as `-instanceName ${instance.name}` parameter.
2. The versions of the migration source and target installations are passed to the product migration API as `-srcVersion` and `-destVersion` parameters when required.
3. The `db.url`, `db.username`, and `db.password` parameters are passed to the product migration API as `-cloneDbUrl`, `-cloneDbUser`, and `-cloneDbPassword`.

If the template includes custom `migrate-xxx-yyy-sbs.dat` migration settings files, the files are passed to the migration utility as `-importFile` parameter.

For detailed information about the product migration utilities, see *Upgrading Software AG Products*.

Step 7

Processes files that are defined inline and applies configuration settings.

Step 8

Restarts the product instances.

Validating a Composite Template

After you run the `apply composite templates` command, Command Central validates the composite template as a first processing step. You can also validate a composite template before applying the template, using ["sagcc exec templates composite validate" on page 468](#).

The validation of the composite template either triggers a new job, or gets executed as part of the main composite template job. In both cases, when you view the job

(from Jobs in the Command Central web user interface), you can access a filtered `default.log` that shows only the validation report entries. If you set the log level of the `com.softwareag.platform.management.client.template.composite.impl.validate` logger to TRACE, the validation report also includes the full composite template with resolved reference values. For information about changing the logging level, see ["Changing the Log Configuration Settings" on page 494](#).

Command Central checks the template validity as follows:

- Tests the connection to the product and fix repositories.
- Verifies the content of the product and fix repositories and checks if the products and fixes defined in the template and their dependencies are available in the respective repositories.
- Tests the connection to the remote nodes and checks if:
 - The remote nodes are accessible over SSH.
 - The authentication credentials for the remote nodes are correct.
 - Platform Manager is already installed.

The validation of a remote node is successful when either the remote node is accessible over SSH, or Platform Manager is installed on the remote node.

- Validates the installation directory. Checks if the installation directory is valid and different from the installation directory of other Platform Managers installed on the same node.
- Writes the validation report to the `default.log`.

Creating a New Environment Using a Composite Template

You use the Command Central command line tool to provision a new environment through a composite template.

To create a new environment through a composite template

1. Open the Command Prompt or shell window and change the working directory to the directory that contains the composite template definition file. For example, C:
`\CommandCentral\templates\myTemplate`
2. Verify that the local Command Central command line tool is running with the following command:

```
sagcc list landscape nodes local -e ONLINE
```
3. Import the template in Command Central using the `sagcc exec templates composite import` command. For example:

```
sagcc exec templates composite import -i template.yaml
```
4. Optional. Verify that your template is now registered in Command Central with the `sagcc list templates composite` command, for example:

```
sagcc list templates composite -e myTemplate
```

5. Apply the template by executing the `sagcc exec templates composite apply` command with the template alias name, for example:

```
sagcc exec templates composite apply myTemplate -i my.properties
```

If the composite template definition contains parameters with variables, you must specify the values for those parameters in the composite template apply command.

You can check the status of the composite template apply job and view the `composite-templates.log` file in the Command Central web user interface or using the jobs and log diagnostic commands of the Command Central command line tool. For more information see, "[Monitoring the Status of a Composite Template](#)" on page 117.

Updating a Provisioned Environment Using the Same Composite Template

You can update an existing environment using the same composite template that you used to create this environment as follows:

1. Use a copy of the template files from your version control system.

Important: Do not modify the template files located under the Command Central profile.

2. Open the `template.yaml` file of the composite template and make the changes that you require.
3. Delete the last version of the composite template that you imported in Command Central, using the `sagcc delete templates composite` command.

Note: You can skip this step, if you run the `sagcc exec templates composite import` command with the `overwrite` parameter set to `true`.
For example: `sagcc exec templates composite import -i template.yaml overwrite=true`

4. Import the updated composite template in Command Central using the `sagcc exec templates composite import` command.
5. Apply the updated composite template with the `sagcc exec templates composite apply` command.

Important: You must always provide a complete set of input parameters to apply the template even if you are re-applying the template against an environment that is partially configured.

If Platform Manager is already bootstrapped and uses a custom administrator password, you must specify the custom password in the composite template `nodes` section. See the custom password example in "[Nodes](#)" on page 88.

Monitoring the Status of a Composite Template

After you run the apply composite template command, you can view the status of a composite template apply job in the Jobs view in the Command Central web user interface. If a sub-job of the main composite template apply job fails:

- In most cases, the sub-job fails with an ERROR status and the main composite template apply job returns a WARNING status.
- The sub-job might fail with an ERROR status, but the main composite template apply job returns a DONE status.
- The failure of the sub-job might cause the main template apply job to fail with an ERROR.

Tip: When you hover over the WARNING or ERROR status of a sub-job, you can view details about the error.

Correcting a Failed Composite Template Apply Operation

When applying a composite template fails, you can take the following steps to correct the failed apply operation:

1. Verify the status of the composite template apply job in the Jobs view in the Command Central web user interface or using the jobs command in the CLI.
2. Check the default.log file in the Command Central web user interface or using the log diagnostic commands in the CLI.
3. Check the logs of the Platform Manager on the target host.
4. Run the apply composite template command again.

2 Developing Composite Templates

■ Creating a Custom Composite Template	120
■ Migrating to a Different Host with Cloned Database	120
■ Migrating Using Disconnected Migration	121
■ Managing Database Components Using a Composite Template	122

Creating a Custom Composite Template

You can design and create a custom template definition file using a text editor with basic YAML validation. Start with the [Creating Command Central Composite Templates](#) tutorial, which walks you through the steps required to develop a basic composite template.

You can also use the sample composite templates from the [sagdevops-9.x-template-samples](#) project on github as a source template or example for creating custom composite templates. Read and follow the instructions in the [readme](#) file of the project and then select and modify the sample composite template that matches or is close to your requirements.

Migrating to a Different Host with Cloned Database

To migrate an environment to a different host with a cloned database, using a composite template:

1. Clone the database you use for the environment to the target host, following the instructions provided by your database vendor.
2. Create a custom composite template of the environment as described in "[Creating a Custom Composite Template](#)" on page 120.
3. In the `templates/products/product_instance` sections, set the database parameters of the product instances to point to the cloned database. For example:

```
templates:
  templateAlias:
    products:
      productID:
        productInstance_name:
          db.url: ${db.cloned.url}
          db.username: ${db.cloned.username}
          db.password: ${db.cloned.password}
```

The database parameters of the product instance in this example template snippet refer to the values of `db.cloned.url`, `db.cloned.username`, and `db.cloned.password`, which you can specify either in the environment/default section of the template definition or in a separate environment properties file. In the following example, the values of the cloned database parameters are set in an environment properties file:

```
db.cloned.url="jdbc:wm:sqlserver://dbhost1:1433;databaseName=db1_CLONED"
db.cloned.username=myDbUser_CLONED
db.cloned.password=myDbpassword_CLONED
```

4. In the migration/nodes section of the template definition, define the host of the source node alias. For example:

```
migration:
  nodes
    envType1:
```

```
node1:
  host: ${src.host}
```

- In the nodes section, define a target host different from the source host for the same node alias. For example:

```
nodes
  envType1:
    node1:
      host: ${other.host}
```

- If you want to use a source archive that already exists on the Command Central source host or on the target host, set the migration section options as follows:

```
migration:
  options:
    backup:
      execute: false
```

When `execute: false`, Command Central searches for the source archive:

- On the Command Central host, in `CommandCentral_directory/profiles/CCE/data/migration/source/migration_source_templateAlias_nodeAlias`
- If Command Central does not find the archive on its own host, it searches on the target host, in `user.home/migration_source_templateAlias_nodeAlias` or at the location that you specify in the `targetPath` parameter of the `backup` option. For example:

```
migration:
  options:
    backup:
      execute: false
      targetPath: C:\sag\myArchive_node1
```

- Import the composite template in Command Central and apply the template.

Migrating Using Disconnected Migration

To perform a disconnected migration, you must create an archive of the source nodes you want to migrate and point Command Central to the location of the archive, so that Command Central can use the source archive to migrate the environment instead of connecting directly to the source nodes.

- Clone the database you use for the environment to the target host, following the instructions provided by your database vendor.
- Create a custom composite template of the environment as described in ["Creating a Custom Composite Template" on page 120](#).
- In the `templates/products/product_instance` sections, set the database parameters of the product instances to point to the cloned database. For example:

```
templates:
  templateAlias:
    products:
      productID:
        productInstance_name:
          db.url: ${db.cloned.url}
```

```
db.username: ${db.cloned.username}
db.password: ${db.cloned.password}
```

The database parameters of the product instance in this example template snippet refer to the values of `db.cloned.url`, `db.cloned.username`, and `db.cloned.password`, which you can specify either in the environment/default section of the template definition or in a separate environment properties file. In the following example, the values of the cloned database parameters are set in an environment properties file:

```
db.cloned.url="jdbc:wm:sqlserver://dbhost1:1433;databaseName=db1_CLONED"
db.cloned.username=myDbUser_CLONED
db.cloned.password=myDbpassword_CLONED
```

4. In the nodes section, define a target host different from the source host for the same node alias. For example:

```
nodes
  envType1:
    node1:
      host: ${other.host}
```

5. If you want to use a source archive that already exists on the Command Central source host or on the target host, set the migration section as follows:

```
migration:
  sourceType: ARCHIVES
  archives:
    targetPath: path/to/source_archive.zip
```

Command Central searches for the source archive:

- a. On the Command Central host, in `<CommandCentral_directory>/profiles/CCE/data/migration/source/<nodeAlias>.zip`
 - b. If Command Central does not find the archive on its own host, it searches on the target host, in `<user.home>/migration_source_<nodeAlias>.zip`
 - c. At the location that you specify in the `targetPath` parameter in the `archives` section.
6. Import the composite template in Command Central and apply the template.

Managing Database Components Using a Composite Template

You can add a database section with database configuration actions in a layer, defined in a composite template definition. When applying the composite template with a database layer, Command Central executes one of the following actions, based on the value of the `environment.mode` parameter in the apply composite template command:

- When `environment.mode=provision`, Command Central creates the database components defined in the database layer
- When `environment.mode=migration`, Command Central upgrades the database components defined in the database layer

Note that the layer in which you include database configuration actions must map to a local or remote Platform Manager node that meets the following requirements:

- The remote Platform Manager node is of release 9.10 or higher.
- The Database Component Configurator is installed on the node, which is usually the case for the local Platform Manager node. You can ensure that the Database Component Configurator is installed by adding an inline template in the composite template definition that will install the Database Component Configurator.

The `databases` layer typically includes one or more `dbAction` sections, where `dbAction` is the name of the action, for example `esb_db`. Each `dbAction` section contains the parameters, listed in the mapping table in ["Supported Database Component Configurator Parameters" on page 125](#) with the following modifications:

<u>This CLI parameter...</u>	<u>is defined in the composite template as...</u>
<code>product</code>	<code>products</code>
<code>component</code>	<code>components</code>

In each `dbAction` section, you can specify only the `components` parameter, only the `products` parameter, or both parameters. If you specify only the `components` parameter, the database action is executed for the database components with the specified code, for example:

```
components: [STR]
```

will execute the action for the database component with code `STR`.

When you have included both the `components` and `products` parameters, Command Central processes first the `components` parameter and then the `products` parameter. You must specify a separate version parameter for the `components` and `products` actions, because the `components` (supported by the Database Component Configurator) and the `products` use different version numbering.

If `version: latest`, you can use the same version parameter for the `components` and `products` included in a single `dbAction` section.

You can list several values in the `components` or `products` parameters. The values in a list are processed in the order they are specified. For example:

```
products: [IS, TN]
```

will execute first the database action for Integration Server and then for Trading Networks.

Examples

- To define an `esb_db` action in the `databases` layer for the database components of the products with IDs `IS` and `TN`, which will get executed on the SQL server at URL `jdbc:wm:sqlserver://DBserver:1433`, for the database user with name `webmuser` and password `webmpass`:

```
layers
  databases:
    esb_db
      description: create db components for IS and TN
      products: [IS,TN]
      db.type: sqlserver
      db.url: "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
      db.username: webmuser
      db.password: webmpass
```

- In the following template snippet, the databases layer includes a “dbc-components” action and a “dbc-products” actions. The parameters included in “dbc-components” and “dbc-products” use by reference the values of the database connection and schema parameters, defined in the environments/default section. The “dbc-components” action will get executed for the database storage component “STR”, and the “dbc-products” action will get executed for the products with IDs “IS”, “BPM”, and “MWS”. Note that the version parameter for each action refers to a different version parameter in the environments/default section. The “dbc-components” action uses the value of the `db.component.version` parameter, and the “dbc-products” action uses the value of the `db.product.version` parameter.

```
environments:
  default:
    db.type:      sqlserver
    db.host:      DBserver
    db.port:      1433
    db.name:      TESTDB
    db.username:  webmuser
    db.password:  webmpass
    db.url:       "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
    db.admin.user:  adminuser
    db.admin.password: adminpass
    db.components: [STR]
    db.products:   [IS,BPM,MWS]
    db.component.version: latest
    db.product.version: 9.10.0.0
  ...
layers:
  databases:
    dbc-components:
      components:    ${db.components}
      version:       ${db.component.version}
      db.type:       ${db.type}
      db.url:        ${db.url}
      db.username:   ${db.username}
      db.password:   ${db.password}
      db.name:       ${db.name}
      db.admin.username:  ${db.admin.username}
      db.admin.password:  ${db.admin.password}

    dbc-products:
      productst:    ${db.products}
      version:      ${db.product.version}
      db.type:      ${db.type}
      db.url:       ${db.url}
      db.username:  ${db.username}
      db.password:  ${db.password}
      db.name:      ${db.name}
```

Supported Database Component Configurator Parameters

The following table maps the Database Component Configurator parameters for configuring database components to the arguments that the Command Central CLI and composite template definition support:

Database Configurator Parameter	CLI or composite template argument
dbms	db.type
component	component
product	product
version	version
user	db.username
password	db.password
dbname	db.name
url	db.url
adminuser	db.admin.user
adminpass	db.admin.password
tablespacedir	db.tablespace.dir
tablespacefordata	db.tablespace.data
tablespaceforindex	db.tablespace.index

For the description and values of the parameters of the Database Component Configurator, see *Installing Software AG Products*.

For the CLI commands syntax and the composite template definition syntax, see *Managing Database Components Using CLI Commands* and ["Managing Database Components Using a Composite Template" on page 122](#).

Database Actions in the Sample Composite Templates

The sample composite templates listed in the table include database action sections that by default create or migrate the database schema for the specified products. The database action that Command Central executes is determined by the `environment.mode` parameter in the apply composite template command, as described in ["Managing Database Components Using a Composite Template" on page 122](#).

The sample templates in the table also include database action parameters that you can customize, including several required parameters without default values.

To view the database action sections and parameters, go to the sample composite templates located at <https://github.com/SoftwareAG/sagdevops-9.x-template-samples>.

Template alias	Parameters	Database component or schema created or migrated
bpms	<code>components:</code> [STR]	Storage and all components supported by the Database Component Configurator.
	<code>products:</code> [IS, BPM, MWS]	Database schemas for Integration Server, My webMethods Server, and Process Engine.
is-layer	<code>components:</code> [STR]	Storage and all components supported by the Database Component Configurator.
	<code>products:</code> [IS]	Database schema for Integration Server.
mws-layer	<code>components:</code> [STR]	Storage and all components supported by the Database Component Configurator.
	<code>products:</code> [MWS]	Database schema for My webMethods Server.

You can also use the sample `dbc` template to create a standalone database schema. For more information about the sample `dbc` template, see ["Summary of Sample Composite Templates" on page 99](#).

3 Using Micro Templates

■ What is a Micro Template?	128
■ When to Use Micro Templates and When a Composite Template?	128
■ Designing Micro Templates	129
■ Using the Sample Micro Templates	129
■ Creating a User-Defined Micro Template	130
■ Validating a Micro Template	131

What is a Micro Template?

You use micro templates to create the layers in a product stack. A product stack is a set of integrated products that work together as one product solution, such as an integration solution or a process and case management solution. A layer in a product stack includes the product instances and run-time components that logically function as a unit.

From the Stacks screen in the Command Central web user interface or using the CLI stacks commands, you can create a product stack with the infrastructure and run-time layers that you require. A layer in a product stack refers to a micro template and uses the properties defined in that template to create and configure the layer. You must define a micro template for each layer that you want to include in the product stack and create a layer type definition that references the micro template. A product stack must have just one infrastructure layer and as many run-time layers as required for the product solution. To create a stack you should define:

- One infrastructure micro template and create a CCE-LAYER-TYPES-INFRA configuration instance that references that template. Based on your use-case, you can define one of the following infrastructure micro templates:
 - Local infrastructure to install the local Platform Manager
 - Remote infrastructure to connect to a remote Platform Manager
 - Remote infrastructure to bootstraps Platform Manager on a remote machine
- Run-time micro template for each functional product unit and create a CCE-LAYER-TYPES-RUNTIME configuration instance that references that template. You can define run-time micro templates for non-clustered and clustered product instances.

In addition, you can create templates that help you when creating layers, for example you can define templates for configuring repositories, licenses, and databases.

Important: Micro templates are used *only* in the layer definitions to create and manage layers with the Command Central Stacks web user interface or the CLI stacks commands. Do not use micro templates with the CLI apply composite templates command.

When to Use Micro Templates and When a Composite Template?

If you use a single composite template to provision a very large environment, the composite template could get very long and complex. In such cases, you can split the composite template into layers, define a micro template for each layer and create a product stack out of the layers to use for provisioning your environment.

A composite template works best when you want to use a single template from which you create different environments, using external environment properties files for each type of environment (development, test, and production).

Designing Micro Templates

Micro templates are defined as a single `template.yaml` file, based on the Command Central DSL template definition. Each micro template is stored in a separate folder that contains a template definition file, named "template.yaml". You can store the user-defined micro templates in a version control system to be able to test and re-use the templates. To design a micro template, you first determine whether the template will be used to create an infrastructure layer or a run-time layer. Templates that you use to install and configure Platform Manager are infrastructure micro templates. Templates that you use to install and configure products (and their hosted run-time components) are run-time micro templates. Both types of micro templates include the following template definition sections:

```
environments
layers
templates
provision
```

In addition to the listed sections, the infrastructure micro templates include a `nodes` section. Run-time micro templates do not have a `nodes` section.

Each micro template (infrastructure or runtime) should include *only* one product inline template that has only one product instance.

In the `environments` section of the micro templates, you can define the following default parameters that are used for creating the layers:

```
environments:
  default:
    nodes: ${}           # REQUIRED A comma-separated list of hostnames
                        # on which to provision the layer: host1,host2
    release:             # The release version of the stack as specified
                        # at the time of creating the stack.
    repo.product:       # The product repository to use for installing products.
    repo.fix:           # The fix repository to use for installing fixes.
    repo.asset:         # The asset repository to use for installing assets.
    cc.installer:       # The bootstrap installer to use when bootstrapping
                        # Platform Manager when creating an infrastructure layer.
```

See also the *Template Definition Reference* topics in the Command Central online help.

Using the Sample Micro Templates

You can start creating layers to build stacks using the sample micro templates available in the [sagdevops-templates.git](https://github.com/sagdevops-templates) project.

To use the sample templates, you must first set up Command Central and import the sample micro templates as described in the readme file of the [sagdevops-templates.git](#) project.

Important: Micro templates are used only in the layer definitions to create and manage layers with the Command Central Stacks web user interface or the CLI ["stacks" on page 449](#) commands. Do not use micro templates with the CLI [apply composite templates](#) command.

Creating a User-Defined Micro Template

Before you can use micro templates, you must install and configure Command Central as described in this help. You can also use the [sagdevops-cc-server](#) project to automate setting up Command Central.

1. Define the micro template YAML definition. See ["Designing Micro Templates" on page 129](#).

You can model the user-defined micro template after a sample micro template, for example see the samples in the [sagdevops-templates.git](#) project.

2. Import the user-defined micro templates in Command Central using the ["sagcc exec templates composite import" on page 467](#) command.
3. Create an instance of the CCE-LAYER-TYPES configuration type that references the user-defined micro template with the ["sagcc create configuration data" on page 213](#) command.

Example:

```
sagcc create configuration data sag01 OSGI-CCE-ENGINE CCE-LAYER-TYPES
--input c:\inputs\layer_data.xml --password secret
```

The input file specifies layer details, such as the name of the layer type instance, the type of the layer, and the alias of the micro template to which the layer refers. You can also specify if the layer is scalable, that is if you can add more nodes after creating the layer.

To determine how to specify the data in the input file, use ["sagcc get configuration data" on page 218](#) to retrieve data for the same type of configuration instance you want to create. For example, run the following command: `sagcc get configuration data local OSGI-CCE-ENGINE RUNTIME-EXISTING --format xml` to get the configuration data for the CCE-LAYER-TYPES-runtime-existing configuration instance.

You can also use a micro template to define the layer type configuration instances. See the sample layer definitions template at <https://github.com/SoftwareAG/sagdevops-templates/tree/master/templates/sag-layer-defs>

After creating the micro templates, you can use them to create layers and add them in a product stack using:

- The Stacks screen in the Command Central web user interface. See ["Create a Software Stack"](#) on page 55.
- The CLI stacks and layers commands. See ["sagcc create stacks"](#) on page 448 and ["sagcc create stacks layers"](#) on page 449.

Validating a Micro Template

To validate a micro template, you can:

- Create a layer that references the micro-template you want to validate and check the layer provisioning jobs in the Jobs view in the Command Central web user interface.
- After importing the template in Command Central, run the ["sagcc exec templates composite validate"](#) on page 468 command with the alias of the micro template and check the validate command jobs in the Jobs view.

X Command Central Developer Reference

■ Using the Command Line Interface	135
■ Managing Database Components Using CLI Commands	157
■ Options for the Commands	165
■ Administration Commands	197
■ License Inventory Commands	203
■ Configuration Commands	209
■ Diagnostics Logs Commands	241
■ Instance Management Commands	249
■ Inventory Commands	257
■ Jobmanager Jobs Commands	291
■ Landscape Commands	297
■ License Keys Commands	319
■ License Reports Commands	329
■ Lifecycle Commands	355
■ Monitoring Commands	361
■ Provisioning Assets Commands	369
■ Provisioning Bootstrap Installers Commands	375
■ Provisioning Fixes Commands	385
■ Provisioning Products Commands	389
■ Repository Commands	393
■ Resources Commands	435
■ Security Credentials Commands	437
■ Stacks and Layers Commands	447
■ Template Commands	463
■ Configuring Command Central and Platform Manager	471
■ Using the Command Central and Platform Manager Logs	489

■ Introduction to Command Central REST API	499
--	-----

4 Using the Command Line Interface

■ Installing CLI as a Remote Client	136
■ Displaying Help for the Command Line Interface	137
■ Executing Command Central Commands	137
■ Executing Platform Manager Commands	138
■ Getting Familiar with Using Commands	138
■ Return Codes from Command Execution	140
■ Creating Shell Scripts that Execute Commands	140
■ Creating Ant Scripts that Execute Commands	141
■ Parameters to Use with the ccsetup Task	141
■ Parameters to Use with the cc Task	143
■ Using a Unix Shell Script to Change the Administrator Password for Command Central	155

Important: The command-line interface in Command Central and Platform Manager version 9.8 and above might not be fully compatible with earlier versions. To use version 9.8 and above, you might need to make changes to the scripts that you developed with earlier versions.

Installing CLI as a Remote Client

You can install CLI separately on a machine other than the Command Central host machine so you can run Command Central commands remotely.

1. Log on to the machine on which to install. No special privileges are required to install CLI.
2. Open your installation email from Software AG and follow the instructions to download the Command Central bootstrapper for the appropriate operating system. If you are installing on a Windows system, extract the ZIP archive into the target directory.
3. Open a command window as Administrator for Windows or a shell window for UNIX and run the .bat file or .sh script with the arguments in the table below.

Argument	Value
-D <i>component</i>	Specify CLI.
-H <i>host_name</i>	DNS name or IP address for the Command Central host machine.
-d <i>path</i>	Full path to the directory in which to install CLI.
-c <i>port_number</i>	HTTP port used by Command Central. The default is 9080.
-C <i>port_number</i>	HTTPS port used by Command Central. The default is 9081.
-p <i>password</i>	Administrator password for Command Central.

4. If you want to change these settings later, rerun the bootstrapper with different values.

Example

- To install on a Windows system and configure CLI to point to Command Central:

```
cc-def-9.12-release-w64.bat -d C:\AdminTools\sagcc -D CLI -H cchost.com
```

```
-c 9100 -C 9101 -p manage123
```

Displaying Help for the Command Line Interface

Important: Beginning with Command Central release 9.12, the `cc` ID of the Command Central command line interface, which is used in the command syntax, is deprecated and replaced by `sagcc`.

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 `sagcc` with no other arguments. For example:

```
sagcc
```

- 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:

```
sagcc --help --server http://rubicon:8090/ccc
```

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:

```
sagcc --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.

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
```

2. Enter the command you want to execute.

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

```
sagcc 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
```

2. Enter the command you want to execute, using the `{--server | -s}` option to identify the Platform Manager server. For more information, see ["server" on page 189](#).

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

```
sagcc 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.

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 the section about landscape commands in this help.

Step	Description	Examples
1.	Use a <code>list</code> command to view the type of information the command returns.	<p>Execute the following command to view a list of installations.</p> <pre>sagcc list landscape nodes</pre> <p>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 installation, execute actions against an installation, or delete an installation.</p>
2.	Use a <code>get</code> or <code>list</code> command to retrieve information for a specific instance.	<p>In this example, assume the <code>list</code> command provided information for an installation that has the alias name</p>

Step	Description	Examples
	<p>Note The <code>get</code> and <code>list</code> commands are equivalent.</p>	<p>“sag01”. To retrieve information for the “sag01” installation, returning the information to an output file in XML format, execute the following command:</p> <pre>sagcc get landscape nodes sag01 --output info --format xml</pre>
3.	<p>Use a <code>create</code> command to create a new instance.</p> <p>You can edit the output file that a <code>get</code> command returns to specify the information for the new instance. Then you can use that file as input to the <code>create</code> command.</p>	<p>To create a new installation with alias name “new”, edit the <code>info.xml</code> file that the <code>get</code> command returned to supply the alias name, URL, and description for the new installation. Then execute the following command:</p> <pre>sagcc create landscape nodes --input info.xml</pre> <p>Note If you execute the <code>list</code> command again, the command lists the “new” installation.</p>
4.	<p>Use an <code>update</code> command to update data for an instance.</p> <p>You can use a <code>get</code> 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 <code>get</code> command returns and use that as the input to the <code>update</code> command.</p>	<p>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:</p> <pre>sagcc get landscape nodes new --output updatefile --format xml</pre> <p>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:</p> <pre>sagcc update landscape nodes new --input updatefile</pre>
5.	<p>Use an <code>exec</code> command to execute an action against an instance.</p>	<p>To generate a new ID for the “new” installation, execute the following command:</p> <pre>sagcc exec landscape nodes new generateNodeId</pre>
6.	<p>Use a <code>delete</code> command to remove an instance.</p>	<p>To delete the “new” installation, execute the following command:</p> <pre>sagcc exec landscape nodes new</pre>

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.

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 <code>{--expected-values -e}</code> option.
<i>response-code</i>	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. <div data-bbox="493 1205 1365 1409" style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>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 <i>response-code</i> modulo 256.</p> </div>

Creating Shell Scripts that Execute Commands

On Windows, execute `sagcc` 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 sagcc 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 sagcc 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 sagcc 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/cee`
-

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

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.

Parameter and Description

- 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:

- Value set in the custom `cc.properties` file located in the `user_home \.sag` directory.
- 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:

- Value set in the custom `cc.properties` file located in the `user_home \.sag` directory.
- 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"
```

Parameter and Description

```
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:

```
sagcc list landscape nodes
```

In a script, use the following:

```
<cc command="sagcc list landscape nodes" />
```

Another example might be to execute the following command:

```
sagcc create landscape nodes alias=n1 url=localhost
```

In a script, use the following:

```
<cc command="sagcc create landscape nodes alias=n1
url=localhost" />
```

Note: Do not include the command options as described in "[Common Options](#)" on [page 167](#). 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:

```
sagcc create landscape nodes alias=n1 url=localhost --format json
```

In a script, use the following:

```
<cc command="sagcc create landscape nodes alias=n1 url=localhost"
format="json"/>
```

`debug`

Parameter and Description

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"
```

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"
```

Parameter and Description

Use the `expectedvalues` parameter in conjunction with the `checkevery` and `wait` parameters.

Tip: Using `wait="0"` with `expectedvalues` acts as a 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`)
- Comma-separated values (`csv`)
- JavaScript Object Notation (`json`)
- ZIP (`zip`)
- PDF (`pdf`)

If you omit the `format` parameter, the command uses `xml`.

Parameter and Description

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:

- 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:

Parameter and Description

```
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.

The logged results include:

- Service output
- Errors that occur while interpreting a command

Parameter and Description

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.
- PUT or UPDATE to update data for a resource.
- DELETE or REMOVE to delete data.
- OPTIONS or WADL to retrieve information for supported services.

Parameter and Description

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.

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:

```
sagcc 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.

Parameter and Description

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"
```

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.

Parameter and Description

`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

`retry`

Optional. Use only in conjunction with `syncJob`. Specifies the number of times to resubmit the command when Command Central gets restarted during a command operation and does not respond within the time interval specified in the `waitForCC` option. If `waitForCC` is specified with no value, the command uses the value of the `CC_WAIT` environment variable, or 120 seconds if `CC_WAIT` is not specified.

For example:

```
retry="1"
```

If you do not include `retry` or you specify `retry="0"`, the command will not be resubmitted.

Parameter and Description

Use `retry`, `syncJob` and `waitForCC` to execute commands with long-running operations more reliably.

`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"
```

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.

`syncJob`

Required if `retry` is used. When you use this option, Command Central monitors and reports progress details while the job is running, and returns the job status and status description after the job completes. When you omit this option, Command Central does not monitor and report the job progress. To enable job monitoring:

```
syncJob="true"
```

Even if `waitForCC` is not included in your script, specifying `syncJob` automatically triggers `waitForCC`. If you use only `syncJob` and omit `waitForCC`, the command

Parameter and Description

uses the value of the `CC_WAIT` environment variable, or 120 seconds if `CC_WAIT` is not specified.

If you execute a command with `syncJob` and `expectedValues`, the result that the command returns is verified against the specified expected values.

Use `syncJob`, `retry` and `waitForCC` to execute commands with long-running operations more reliably. This will ensure that the command is resubmitted in case Command Central is restarted during the operation.

`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 command uses the value of the `CC_WAIT` environment variable. If the `CC_WAIT` environment variable is not set, the command uses 120 seconds.

`waitForCC`

Optional. Specifies the time interval in seconds the command waits for Command Central to come online after starting or restarting Command Central. If you omit this option, the command fails when Command Central is not online. For example:

```
waitForCC="60"
```

Both `wait` and `waitForCC` use the value of the `CC_WAIT` environment variable. Use `waitForCC` in conjunction with `wait` if you specify a value for `waitForCC` higher than 120 seconds. In this case, specify a value for `wait` higher than the value of `waitForCC`.

When you monitor a long-running job, you can use `waitForCC` and `syncJob` to continue monitoring the job even after Command Central is restarted.

Parameter and Description

Using `waitForCC` is most helpful with commands that trigger long-running jobs, for example the `apply composite templates` command. However, you can use `waitForCC` with any command.

Using a Unix Shell Script to Change the Administrator Password for Command Central

You can use the following sample Unix shell script to change the Administrator user password for Command Central.

```
NODE_ALIAS=local
USERNAME=Administrator
PASSWORD=manage123
RCID=OSGI-CCE

sagcc get configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS
-Administrator -o administrator.xml
sed "s,/>, ><Password>${PASSWORD}</Password>
</User>,g" administrator.xml > administrator_new.xml

sagcc update configuration data $NODE_ALIAS $RCID COMMON-LOCAL-USERS
-Administrator -i administrator_new.xml

sed "s,^password=.*$,password=${PASSWORD},g" $HOME/.sag/cc.properties >
cc.properties
cp cc.properties $HOME/.sag/cc.properties
sagcc get landscape nodes
```


5 Managing Database Components Using CLI Commands

■ Supported Database Component Configurator Actions	158
■ Create Database Components	158
■ Drop Database Components	159
■ Migrate Database Components	160
■ Recreate Database Components	161
■ Execute a Database Action with Catalog	161
■ Searching for Database Component Configurator Details	162

Supported Database Component Configurator Actions

With the Command Central CLI commands, you can perform a database action on a database component, or for the database components of one product. The following topics provide examples of the CLI commands for each database configuration action.

You can use the Command Central CLI to perform the following actions with a database component:

- create
- drop
- recreate
- migrate
- list catalog or other details

Before you use the CLI commands to perform any of the database actions, see ["Before Creating Database Components" on page 44](#).

For a description and additional information about the create, drop, recreate, or list existing database components actions, see *Installing Software AG Products*.

For details and instructions about when to migrate database components, see *Upgrading Software AG Products*.

Create Database Components

Command Syntax

- To create a database component:

```
sagcc exec administration product local DatabaseComponentConfigurator
database create db.type=db_type component=componentID version=db_version
db.username=db_username db.password=db_password db.name=db_name
db.url=db_url db.admin.username=db_admin_username
db.admin.password=db_admin_password
db.tablespace.dir=db_tablespace_dir
db.tablespace.data=data_tablespace_name
db.tablespace.index=index_tablespace_name
```

- To create the database components for a product:

```
sagcc exec administration product local DatabaseComponentConfigurator
database create db.type=db_type product=productID
version=db_version db.username=db_username
db.password=db_password db.name=db_name db.url=db_url
```

Usage Notes

The value of the `version` parameter for database components is specific for each component. Software AG recommends that you specify `version=latest` to create the latest version of a database component.

The value of the `version` parameter for a product is the four number sequence of the product release version, for example `9.10.0.0`. You can also specify `latest` as the `version` value in a product command, in which case the configurator will create database components for the latest release version of the product.

Examples

- To create a database component in the local Platform Manager installation for the component with code "STR" and version "latest", on the SQL server at URL "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB", for the database user "webmuser" with password "webmpass", and the specified operating system administrator:

```
sagcc exec administration product local DatabaseComponentConfigurator
database create db.type=sqlserver component=STR
version=latest db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
db.admin.username=adminuser db.admin.password=adminpass
```

- To create a database with name "TESTDB" in the local Platform Manager installation for the product with ID "IS" and version "9.10.0.0", on the SQL server at URL "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB", and for the database user "webmuser" with password "webmpass":

```
sagcc exec administration product local DatabaseComponentConfigurator
database create db.type=sqlserver product=IS
version=9.10.0.0 db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

Drop Database Components

Command Syntax

- To drop a database component:

```
sagcc exec administration product local DatabaseComponentConfigurator
database drop db.type=db_type component=componentID version=db_version
db.username=db_username db.password=db_password db.name=db_name
db.url=db_url db.admin.username=db_admin_username
db.admin.password=db_admin_password
db.tablespace.dir=db_tablespace_dir
db.tablespace.data=data_tablespace_name
db.tablespace.index=index_tablespace_name
```

- To drop the database components for a product:

```
sagcc exec administration product local DatabaseComponentConfigurator
database drop db.type=db_type product=productID
```

```
version=db_version db.username=db_username
db.password=db_password db.name=db_name db.url=db_url
db.admin.username=db_admin_username db.admin.password=db_admin_password
```

Examples

- To drop a database component in the local Platform Manager installation for the component with code "STR" and version "latest", on the SQL server at URL "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB", for the database user "webmuser" with password "webmpass", and the specified operating system administrator:

```
sagcc exec administration product local DatabaseComponentConfigurator
database drop db.type=sqlserver component=STR
version=latest db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
db.admin.username=adminuser db.admin.password=adminpass
```

- To drop a database with name "TESTDB" in the local Platform Manager installation for the product with ID "IS" and version "9.10.0.0", on the SQL server at URL "jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB", for the database user "webmuser" with password "webmpass", and the specified operating system administrator:

```
sagcc exec administration product local DatabaseComponentConfigurator
database drop db.type=sqlserver product=IS
version=9.10.0.0 db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
db.admin.username=adminuser db.admin.password=adminpass
```

Migrate Database Components

Important: The CLI commands in this section must be run in the context and order documented in *Upgrading Software AG Products*. Otherwise, you will experience unpredictable results, possibly including corruption of your installation and data.

- To migrate the database components for a product:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=db_type
{product|component}=productID|componentID
version=dest_version db.username=db_username db.password=db_password
db.name=db_name db.url=db_url
```

- To migrate the database components for a product from a specific version:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=db_type
{product|component}=productID|componentID
version=dest_version fromversion=src_version db.username=db_username
db.password=db_password db.name=db_name db.url=db_url
```

Example

- To upgrade the database components for the product with ID “IS” from version “9.9.0.0” to the latest version, on the SQL server at URL “jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB”, and for the database user “webmuser” with password “webmpass”:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=sqlserver product=IS version=latest
fromversion=9.9.0.0 db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

- To upgrade the database component with code “STR” to the latest version, on the SQL server at URL “jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB”, and for the database user “webmuser” with password “webmpass”:

```
sagcc exec administration product local DatabaseComponentConfigurator
database migrate db.type=sqlserver component=STR version=latest
db.username=webmuser db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

Recreate Database Components

Command Syntax

To recreate the database components for a product:

```
sagcc exec administration product local DatabaseComponentConfigurator
database recreate db.type=db_type product=productID version=db_version
db.username=db_username db.password=db_password db.name=db_name db.url=db_url
```

Example

To recreate the database components for the product with ID “IS” and version “9.10.0.0”, on the SQL server at URL “jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB”, and for the database user “webmuser” with password “webmpass”:

```
sagcc exec administration product local DatabaseComponentConfigurator
database recreate db.type=sqlserver product=IS version=9.10.0.0
db.username=webmuser db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
```

Execute a Database Action with Catalog

You can create, drop, or upgrade the database components for a product and list the existing database components.

Command Syntax

```
sagcc exec administration product local DatabaseComponentConfigurator
database {create|drop|migrate} db.type=db_type product=productID
version=db_version db.username=db_username db.password=db_password
db.name=db_name db.url=db_url db.admin.username=db_admin_username
```

```
db.admin.password=db_admin_password catalog=true
```

Example

To create a database with name “TESTDB” with a catalog in the local Platform Manager installation for the product with ID “IS” and version “9.10.0.0”, on the SQL server at URL “jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB”, and for the database user “webmuser” with password “webmpass”:

```
sagcc exec administration product local DatabaseComponentConfigurator
database create db.type=sqlserver product=IS
version=9.10.0.0 db.username=webmuser
db.password=webmpass db.name=TESTDB
db.url="jdbc:wm:sqlserver://DBserver:1433;databaseName=TESTDB"
catalog=true
```

Searching for Database Component Configurator Details

You can use the [sagcc list administration](#) command to search for information about the Database Component Configurator. The details that you can list include:

- Catalog
- Actions
- Database types
- Products
- Components
- Supported upgrade versions

Command Syntax

- To list the catalog of created database components for a product:

```
sagcc list administration product local DatabaseComponentConfigurator
database catalog db.type=db_type db.username=db_username
db.password=db_password db.name=db_name db.url=db_url
```

- To list the actions supported by the Database Component Configurator:

```
sagcc list administration product local DatabaseComponentConfigurator
database actions
```

- To list the the RDBMSs the Database Component Configurator supports:

```
sagcc list administration product local DatabaseComponentConfigurator
database dbtypes
```

- To list the IDs of supported products:

```
sagcc list administration product local DatabaseComponentConfigurator
database products
```

- To list the codes of supported database components:

```
sagcc list administration product local DatabaseComponentConfigurator
database components
```

- To list the supported migration versions:

```
sagcc list administration product local DatabaseComponentConfigurator  
database migrations db.type=db_type  
{product|component}=productID | componentID
```

Example

To list the supported migration versions on the SQL server database for the “MWS” runtime component:

```
sagcc list administration product local DatabaseComponentConfigurator  
database migrations db.type=sqlserver component=MWS
```


6 Options for the Commands

■ Common Options	167
■ accept	167
■ check-every	168
■ configuration-file	170
■ debug	171
■ error	172
■ expected-values	173
■ force	175
■ format	175
■ input	177
■ input-format	179
■ log	180
■ media-type	181
■ output	182
■ output-format	183
■ password	186
■ properties	186
■ quiet	187
■ retry	188
■ server	189
■ ssl-truststore-file	190
■ ssl-trust-all-hosts	191
■ ssl-truststore-password	191
■ sync-job	192
■ username	192
■ wait	193

■ wait-for-cc 194

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.

- "accept" on page 167
- "check-every" on page 168
- "configuration-file" on page 170
- "debug" on page 171
- "error" on page 172
- "expected-values" on page 173
- "force" on page 175
- "format" on page 175
- "input" on page 177
- "input-format" on page 179
- "log" on page 180
- "media-type" on page 181
- "output" on page 182
- "output-format" on page 183
- "password" on page 186
- "properties" on page 186
- "quiet" on page 187
- "server" on page 189
- "ssl-truststore-file" on page 190
- "ssl-trust-all-hosts" on page 191
- "ssl-truststore-password" on page 191
- "username" on page 192
- "wait" on page 193

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

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
<code>content_type</code>	<p>Specifies a well-formed content type that indicates the format you want the command to use for the output. The following lists some examples:</p> <ul style="list-style-type: none"> ■ 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" on page 177](#).
- 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" on page 182](#).

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

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

Syntax

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

Arguments

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

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" on page 173](#).

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

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 485](#).

Example

To execute the command with the Command Central default configuration file:

```
--configuration-file
Software AG_directory
\CommandCentral\client\conf\cc.properties
```

debug

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.

```
sagcc list landscape nodes --debug
```

```

Debug information Request: GET http://localhost:8090/cce/landscape/nodes
Host: localhost:8090
Content-Type: text/tab-separated-values,text/plain,application/xml;q=0.9,*/*;q=0.8
Accept: text/tab-separated-values,text/plain,application/xml;q=0.9,*/*;q=0.8
Response: 200 OK
Content-Type: text/tab-separated-values
Content-Length: 89
Service Output Alias Name Status Url Host Url Port
node125 Name of node node125 ONLINE localhost 8202 ]]]]

```

error

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
<i>file</i>	<p>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:</p> <ul style="list-style-type: none"> ■ 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 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

Specifies the expected values for which to wait before a command completes. Use in conjunction with the [check-every](#) and [wait](#) 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
<i>values</i>	Specifies the values that must be present in the output for a command to complete. Use a comma to separate each value. Always place quotes around the value, for example: <pre>--expected-values "STOPPED,ONLINE"</pre> <p>The values can include the wildcard character (*) that represents any number of characters. For example, <code>RUNN*</code>, <code>*NLINE</code>, <code>OFF*NE</code>.</p>

Operators

The command supports the following operators listed in the table by operator precedence:

Operator	Description
!	Negates the argument to the right, for example <code>!RUNNING</code> returns all responses that do not contain the text “RUNNING”.
,	Specifies the sequence in which the values will get executed. For example, if you specify <code>STOPPED, UNKNOWN, ONLINE</code> , the

Operator	Description
	<p>command first checks for the <code>STOPPED</code> run-time status. After <code>STOPPED</code>, the command checks for the <code>UNKNOWN</code> status, and after that it checks for <code>ONLINE</code>. If any of the run-time statuses do not occur before the command times out, the command returns an error indicating the missing statuses.</p> <p>When you specify multiple values, the command checks for <i>all</i> values and completes successfully only if <i>all</i> of them are present.</p>
<code>&</code>	<p>Indicates that the value includes a logical AND operator. For example, <code>RUNNING&RESPONSIVE</code> checks if the output contains both <code>RUNNING</code> and <code>RESPONSIVE</code>.</p>
<code> </code>	<p>Indicates that the value includes a logical OR operator, for example the following command checks if the output contains either <code>DONE</code> or <code>WARNING</code>:</p> <pre>--expected "DONE WARNING"</pre>

Usage Notes

- You can define the expected values as a complex expression that contains more than one operator. The order of the operations is controlled by using brackets, for example: `"!RU*NG|OFFLINE&!ONLINE"` and `"(!RU*NG) |(OFFLINE&(!ONLINE))"` follow the same order of operations, but `"(! (RU*NG|OFFLINE)) & (!ONLINE)"` follows a different operator precedence.
- If a command is executed from a shell on Windows or UNIX operating systems, the exclamation mark is considered a special character. Use the following escape characters to escape (!):
 - (^) on Windows, for example `"^!RUNNING"`
 - (\) on UNIX/Linux, for example `"\!RUNNING"`
- 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 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 Notes

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

format

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
<code>tsv args</code>	Specifies you want the output in tab-separated values format. For more information about the arguments you can specify, see "properties" on page 186 .
<code>text</code>	Specifies you want the output in plain text format.
<code>xml</code>	Specifies you want the output in XML format.
<code>csv args</code>	Specifies you want the output in comma-separated values format. For more information about the arguments you can specify, see "properties" on page 186 .
<code>json</code>	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" on page 167](#).
- 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" on page 177](#).
- 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" on page 182](#).
- 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 ["properties" on page 186](#).

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
```

input

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

For example, when using the `sagcc 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
<code>filename { .xml .json .properties }</code>	<p>Specifies the file that contains the input data. The input file can be:</p> <ul style="list-style-type: none"> ■ An <code>.xml</code> file containing input data in XML format ■ A <code>.json</code> file containing input data in JavaScript Object Notation format ■ A <code>.properties</code> file containing input data in key/value pairs format. <p>When identifying the input file, you can specify:</p> <ul style="list-style-type: none"> ■ Absolute directory path and filename.

Argument	Description
	<ul style="list-style-type: none"> ■ 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 `sagcc 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 `sagcc create configuration data` command to create a COMMON-PORTS configuration instance, first use the `sagcc 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

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
<i>content-type</i>	<p>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:</p> <ul style="list-style-type: none"> ■ <code>application/xml xml</code> ■ <code>application/json json</code> ■ <code>text/plain text</code>

Argument	Description
	<ul style="list-style-type: none"> ■ text/tab-separated-values tsv ■ text/csv csv <p>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.</p> <p>For the content types that a command supports, see the documentation for a specific command.</p>

Example

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

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

log

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
<i>file</i>	<p>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:</p> <ul style="list-style-type: none"> ■ Absolute directory path and filename.

Argument	Description
	<ul style="list-style-type: none"> ■ 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

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
<code>content-type</code>	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

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
<code>file</code>	<p>Specifies the file where you want the output written. If the file you specify does not exist, the command creates it. You can specify:</p> <ul style="list-style-type: none"> ■ 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 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 writes 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

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
<i>content-type</i>	<p>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:</p> <ul style="list-style-type: none"> ■ application/xml xml ■ application/json json ■ text/plain text ■ text/tab-separated-values tsv ■ text/csv csv <p>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.</p>

Argument	Description
----------	-------------

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
```

Recommendations When Selecting a Content Type

The following table gives some general guidelines to help you determine which of the output content types supported by Command Central to use with different types of clients.

Type of client	Recommended format	Considerations
Web client	JSON	<p>JSON is supported by all browsers.</p> <p>JSON does not guarantee the order of the properties, because during processing JSON uses the names of the properties and their order is not relevant.</p> <p>JSON is handled transparently by web clients when new properties are added by the server.</p>
Non-web API client	XMLJSON	<p>Non-web API clients should use either XML, or JSON, based on which content type is better supported in the programming language or platform used by the API client.</p> <p>The order of the properties in XML is guaranteed, because the order is important during XML processing.</p> <p>With XML and JSON, the API clients must not fail when unexpected fields are detected and unexpected fields must get ignored.</p> <p>The XML and JSON output by default include all properties.</p>
Shell script	TSVCSV	<p>The TSV or CSV content types are well-supported by UNIX (and to some extent Windows) command line shells.</p>

Type of client	Recommended format	Considerations
		<p data-bbox="711 359 1367 474">Note: New command line shells, for example Windows PowerShell, can also process the XML and JSON content types.</p> <p data-bbox="711 495 1334 600">With the TSV and CSV content types, the property names remain unchanged, but the order and the set of properties may change.</p> <p data-bbox="711 621 1292 894">To prevent errors because of changes between releases, the shell script must not depend on the default set and order of properties, but the script must request a specific set and order of properties. To get a specific set of properties in a particular order, the scripting client must request the properties by name, using the <code>properties=propName,propName2</code> parameter.</p> <p data-bbox="711 915 1302 1020">The scripting clients can also remove the header line, using the <code>includeHeaders=false</code> parameter.</p> <p data-bbox="711 1041 1367 1188">Tip: To list all properties supported by the current version of the Command Central server, use the <code>includeHeaders=properties</code> and <code>properties=*</code> parameters.</p>
CLI end user	TSV	<p data-bbox="711 1230 1334 1398">TSV is the default content type when a CLI end-user interaction is involved. With this format, the default set of properties fits into the width of the console and makes the output easy to read for humans.</p> <p data-bbox="711 1419 1334 1671">The default set and order of the properties is not guaranteed, because when properties are added or modified, the order of the columns may change to make the output more user-friendly. However, the TSV content type enables you to determine which properties to show in the output.</p> <p data-bbox="711 1692 1269 1818">By design, the TSV output does not include all object properties, because it is not always possible to map a multi-dimensional object model into a two-dimensional table model.</p>

password

Specifies the password for the user executing the command.

Syntax

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

Arguments

None.

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
```

properties

You can include the `properties` argument to customize the output of a command when the output is in the tab-separated values (tsv) or comma-separated values (csv) format.

Syntax

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

Arguments

Argument	Description
<code>[includeHeaders={labels properties none}]</code>	<p>Specifies whether you want the output to include a header line. Specify one of the following:</p> <ul style="list-style-type: none"> ■ <code>labels</code> 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.

Argument	Description
	<ul style="list-style-type: none"> ■ <code>properties</code> to include a header line containing the property key name for each field. For example “<code>product.version</code>” might be the display name if the output includes the version of a product. ■ <code>none</code> to omit headers from the output.
<pre>[properties=keys]</pre>	<p>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 <code>properties=product.version</code>.</p> <p>To specify multiple keys, separate each with a comma. For example, if you want alias names and descriptions in the output, you might specify <code>properties=alias,description</code>.</p> <p>Use <code>properties=*</code> to include all information. If you omit the <code>properties</code> argument, the command returns a default set of fields.</p>

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 `sagcc list landscape nodes` command to retrieve the list of alias names and descriptions for installations. First, execute the `sagcc 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 `sagcc list landscape nodes` with `--format csv includeHeaders=name properties=alias,description`.

quiet

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.

retry

Use only in conjunction with `{--sync-job | -j}`. Specifies the number of times to resubmit the command when Command Central gets restarted during a CLI command operation and does not respond within the time interval specified in the `{--wait-for-cc | -t}` option.

Syntax

```
{--retry | -y} [count]
```

Arguments and Options

Argument	Description
[count]	Optional. Specifies the number of times to resubmit a command. If you omit the <i>count</i> argument, by default the command uses the value of the <code>CC_RETRY</code> environment variable. If the <code>CC_RETRY</code> environment variable is not set, the command uses 1.

Usage Notes

- If `{--retry | -y}` is not included in the command syntax or is set to 0, the command will not be resubmitted.
- If you specify a number for the `{--retry | -y}` option other than 0, the command waits for Command Central to come online for as many seconds as specified by the `{--wait-for-cc | -t}` option. If `{--wait-for-cc | -t}` is specified with no value, the command uses the value of the `CC_WAIT` environment variable, or 120 seconds if `CC_WAIT` is not specified. For more information, see ["sync-job" on page 192](#) and ["wait-for-cc" on page 194](#).
- Use in conjunction with the `{--sync-job | -j}` and `{--wait-for-cc | -t}` options to execute commands with long-running operations more reliably. For an example, see the "Examples" section of the ["sagcc exec templates composite apply" on page 464](#).

- You can also set `retry` as a parameter of the `cc` tasks included in an Ant script that executes CLI commands. See ["Parameters to Use with the `cc` Task" on page 143](#).

Examples

- To enable job monitoring and resubmit the command three times when Command Central is restarted and comes online within 120 seconds or the value of `CC_WAIT`:

```
--sync-job --retry 3 --wait-for-cc
```

server

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

Syntax

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

Arguments

Argument	Description
<code>url</code>	<p>Identifies the URL of a Command Central or Platform Manager server on which to execute the command.</p> <ul style="list-style-type: none"> ■ When specifying the URL of a Command Central server, if you omit: <ul style="list-style-type: none"> ■ Protocol, the command uses <code>"https://"</code>. ■ Port number, the command uses <code>"8090"</code>, which is the default port for a Command Central server. ■ Context, the command uses <code>"cce"</code>. ■ 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
<i>path</i>	Specifies the fully qualified path to the truststore location. The default Command Central truststore with name demo-truststore.jks is located in the <i>Software AG_directory</i> \CommandCentral\client\conf directory.
	Note: You can specify a relative path to the current execution directory of the sagcc 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 Command Central and Platform Manager 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
```

Arguments

None.

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.

sync-job

When you use this option, Command Central monitors and reports progress details while the job is running, and returns the job status and status description after the job completes. When you omit this option, Command Central does not monitor and report the job progress.

Syntax

```
{--sync-job | -j}
```

Arguments and Options

None.

Usage Notes

- Even if the `{--wait-for-cc | -t}` option is not included in a command, using the `{--sync-job | -j}` option automatically triggers `{--wait-for-cc | -t}`. If you use only `{--sync-job | -j}` and omit `{--wait-for-cc | -t}`, the command uses the value of the `CC_WAIT` environment variable, or 120 seconds if `CC_WAIT` is not specified. For more information, see ["wait-for-cc" on page 194](#).
- If you execute a command with `{--sync-job | -j}` and `{--expected-values | -e}`, the result that the command returns is verified against the specified expected values.
- Use in conjunction with the `{--retry | -y}` and `{--wait-for-cc | -t}` options to execute commands with long-running operations more reliably. This will ensure that the command is resubmitted in case Command Central is restarted during the operation. For an example, see the "Examples" section of the ["sagcc exec templates composite apply" on page 464](#).
- You can also set `syncJob` as a parameter of the `cc` tasks included in an Ant script that executes CLI commands. See ["Parameters to Use with the cc Task" on page 143](#).

Examples

- To enable job monitoring and resubmit the command two times when Command Central is restarted and does not come online for 160 seconds and wait a total of 200 seconds for the whole command to be completed:

```
--sync-job --retry 2 --wait-for-cc 160 --wait 200
```

username

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
<i>user_name</i>	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" on page 186](#)

Example

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

```
--username admin02
```

wait

Specifies how many seconds to wait for a long-running operation to return the expected values. Use in conjunction with the [expected-values](#) and [check-every](#) options.

Syntax

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

Arguments

Argument	Description
<i>seconds</i>	Specifies the number of seconds the command waits for the expected output specified by the <code>{--expected-values -e}</code> option before completing. The default is the value of the <code>CC_WAIT</code> environment variable. If the <code>CC_WAIT</code> 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" on page 173](#).

wait-for-cc

Specifies the time interval in seconds the command waits for Command Central to come online after starting or restarting Command Central. If you omit this option, the command fails when Command Central is not online.

Syntax

```
{--wait-for-cc | -t} [seconds]
```

Arguments and Options

Argument	Description
<code>[<i>seconds</i>]</code>	Optional. Specifies the time interval in seconds the command waits for Command Central to come online. If you do not specify a value for <i>seconds</i> , the command uses the value of the <code>CC_WAIT</code> environment variable or 120 seconds if <code>CC_WAIT</code> is not specified.

Usage Notes

- If `--wait-for-cc 0`, the command fails when Command Central is not online.

- Both `{--wait | -w}` and `{--wait-for-cc | -t}` options use the value of the `CC_WAIT` environment variable. Use `{--wait-for-cc | -t}` in conjunction with `{--wait | -w}` if you specify a value for `{--wait-for-cc | -t}` higher than 120 seconds. In this case, specify a value for `{--wait | -w}` higher than the value of `{--wait-for-cc | -t}`.
- When you monitor a long-running job, you can use the `{--wait-for-cc | -t}` and `{--sync-job | -j}` options to continue monitoring the job even after Command Central is restarted. See the example in ["sync-job" on page 192](#).
- This option is most helpful with commands that trigger long-running jobs, for example the `apply composite templates` command. However, you can use the option with any command. In the following example, the command attempts to retrieve the list of products installed in the local installation and if Command Central does not respond, the command checks the status of Command Central and waits for Command Central to be online for 60 seconds:

```
sagcc list inventory products local --wait-for-cc 60
```
- You can also set `waitForCC` as a parameter of the `cc` tasks included in an Ant script that executes CLI commands. See ["Parameters to Use with the cc Task" on page 143](#).

Examples

- To wait for 60 seconds for Command Central to come online:

```
--wait-for-cc 60
```


7 Administration Commands

■ About Administration Commands	198
---------------------------------------	-----

About Administration Commands

You can run administration commands in the Command Central command line tool to perform custom administrative tasks for managed products, for example create a backup of the file system or a database.

sagcc exec administration

Executes a custom administrative action against a product. The executed action is product specific. For more information about the custom administrative actions supported for a product, see the product-specific topic in this help or the respective product documentation.

Syntax

- Command Central syntax:

- To execute an action against a product:

```
sagcc exec administration product node_alias productid namespace
  administrative_action
[argName1=value1] [argName2=value2]...
[options]
```

- To execute an action against a run-time component:

```
sagcc exec administration component node_alias componentid namespace
  administrative_action
[argName1=value1] [argName2=value2]...
[options]
```

- Platform Manager syntax:

- To execute an action against a product:

```
sagcc exec administration product productid namespace
  administrative_action
[argName1=value1] [argName2=value2]...
[options]
```

- To execute an action against a run-time component:

```
sagcc exec administration component componentid namespace
  administrative_action
[argName1=value1] [argName2=value2]...
[options]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	The alias of the node on which the administrative action is executed.

Argument or Option	Description
<i>productid</i>	The ID of the product for which to execute the administrative action.
<i>componentid</i>	The ID of the run-time component for which to execute the administrative action.
<i>namespace</i>	<p>The namespace identifier for the administrative action. The namespace groups together administration functions that relate to the same domain, such as migration or configuration.</p> <p>If you do not specify a namespace value, Command Central returns a list of all available administration namespaces for the specified product or run-time component.</p>
<i>administrative_action</i>	The administrative action to execute. If you do not specify a value, Command Central returns a list of all available administrative actions for the specified product or run-time component.
[argName1=value1] [argName2=value2]...	Optional. The arguments of the administrative action.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

Based on the executed administrative action, the command returns the output in text, JSON, or XML format.

Examples When Executing on Command Central

- To execute an “admin” action in the “example” namespace, against the local installation for a product with ID “SPM” and use the values specified for “arg1” and “arg2” of the “admin” action:

```
sagcc exec administration product local SPM example adminAction
argument1=val1 arg2=val2
```

- To execute an “admin” action in the “example” namespace, against the local installation for a run-time component with ID “OSGI-SPM” and use the values specified for “arg1” and “arg2” of the “admin” action:

```
sagcc exec administration component local OSGI-SPM example adminAction
```

```
arg1=val1 arg2=val2
```

sagcc list administration

Retrieves a list of available custom administrative actions for a product, such as administration namespaces or functions. For more information about the custom administrative actions that you can list for a product, see the product-specific topic in this help or the respective product documentation.

Syntax

- Command Central syntax:

- To list custom administrative namespaces for a product:

```
sagcc list administration product node_alias productid [namespace]
[administrative_action] [argName1=value1] [argName2=value2]...
[options]
```

- To list custom administrative namespaces for a run-time component:

```
sagcc list administration component node_alias componentid [namespace]
[administrative_action] [argName1=value1] [argName2=value2]...
[options]
```

- Platform Manager syntax:

- To list custom administrative namespaces for a product:

```
sagcc list administration product productid [namespace]
[administrative_action] [argName1=value1] [argName2=value2]...
[options]
```

- To list custom administrative namespaces for a run-time component:

```
sagcc list administration component componentid [namespace]
[administrative_action] [argName1=value1] [argName2=value2]...
[options]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	The alias of the node on which the administrative action is executed.
<i>productid</i>	The ID of the product for which to list custom administrative actions.
<i>componentid</i>	The ID of the run-time component for which to list custom administrative actions.
[<i>namespace</i>]	Optional. The namespace identifier for the administrative action. The namespace groups

Argument or Option	Description
	<p>together administrative functions that relate to the same domain, such as migration or configuration.</p> <p>If you do not specify a namespace value, Command Central returns a list of all available administration namespaces for the specified product or run-time component.</p>
<code>[administration_action]</code>	Optional. The administrative action to execute. If you do not specify a value, Command Central returns a list of all available administrative actions for the specified product or run-time component.
<code>[argName1=value1]</code> <code>[argName2=value2]...</code>	Optional. The arguments of the administrative action.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

Based on the type of custom product assets, the command returns the output in text, JSON, or XML format.

Examples When Executing on Command Central

- To search for the custom administrative actions of a product with ID "SPM" in the "backup" namespace:

```
sagcc list administration product local SPM backup -p manage
```

The product runs on the local installation. The command uses the default user name and specifies "manage" for the user's password.

- To search for the custom administrative actions of a run-time component with ID "OSGI-SPM" in the "backup" namespace:

```
sagcc get administration component local OSGI-SPM backup -p manage
```

The run-time component runs on the local installation. The command uses the default user name and specifies "manage" for the user's password.

8 License Inventory Commands

■ sagcc add inventory components contracts assignment	204
■ sagcc update inventory components contracts assignment	205
■ sagcc delete inventory components contracts assignment	206
■ sagcc get inventory components contracts assignment	206

sagcc add inventory components contracts assignment

Assigns a run-time component to a license.

Syntax

- Command Central syntax:

```
sagcc add inventory components contracts assignment nodeAlias
runtimeComponentId item=contractItemIdentifier [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. Specifies the alias name of the installation in which the run-time component is installed.
<i>runtimeComponentId</i>	Required. Specifies the run-time component to which you want to assign a license.
<i>item=contractItemIdentifier</i>	Required. Specifies the contract item identifier by which a license is defined. A contract item identifier consists of the following components: <ul style="list-style-type: none"> ■ <i>LocationId</i> ■ <i>ContractId</i> ■ <i>ContractItemId</i> ■ <i>BundleComponentId</i>
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To assign the run-time component "IntegrationServer_default" installed in the installation with alias name "sag01" to a license with contract item identifier "0007301234-0002021037-000020-000030":

```
sagcc add inventory components contracts assignment sag01
IntegrationServer_default item=0007301234-0002021037-000020-000030
```

sagcc update inventory components contracts assignment

Updates the license assignment for a run-time component.

Syntax

- Command Central syntax:

```
sagcc update inventory components contracts assignment nodeAlias
runtimeComponentId item=contractItemIdentifier [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. Specifies the alias name of the installation in which the run-time component is installed.
<i>runtimeComponentId</i>	Required. Specifies the run-time component for which to update the assignment of a license.
<i>item=contractItemIdentifier</i>	Required. Specifies the contract item identifier by which a license is defined. A contract item identifier consists of: <ul style="list-style-type: none"> ■ <i>LocationId</i> ■ <i>ContractId</i> ■ <i>ContractItemId</i> ■ <i>BundleComponentId</i>
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To update the license assignment for the run-time component "IntegrationServer_default" installed in the installation with alias name "sag01" to a license with contract item identifier "0007301234-0002021037-000020-000030":

```
sagcc update inventory components contracts assignment sag01
IntegrationServer_default item=0007301234-0002021037-000020-000030
```

sagcc delete inventory components contracts assignment

Deletes the license assignment for a run-time component.

Syntax

- Command Central syntax:

```
sagcc delete inventory components contracts assignment nodeAlias
runtimeComponentId [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. Specifies the alias name of the installation in which the run-time component is installed.
<i>runtimeComponentId</i>	Required. Specifies the run-time component for which to delete the assignment of a license.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To delete the license assignment for the run-time component "IntegrationServer_default" installed in the installation with alias name "sag01":

```
sagcc delete inventory components contracts assignment sag01
IntegrationServer_default
```

sagcc get inventory components contracts assignment

Returns all manual assignments of a run-time component to a license, or a list of assignments based on the following filters:

- Node alias
- Run-time component ID

Syntax

- Command Central syntax:

```
sagcc get inventory components contracts assignment [nodeAlias=nodeAlias]
[runtimeComponentId=runtimeComponentId] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
[<i>nodeAlias=nodeAlias</i>]	Optional. Specifies the alias name of the installation in which the run-time component is installed.
[<i>runtimeComponentId=runtimeComponentId</i>]	Optional. Specifies the run-time component for which to retrieve the manual assignment of a license.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To retrieve the manual license assignment for the run-time component "IntegrationServer_default" installed in the installation with alias name "sag01":

```
sagcc get inventory components contracts assignment nodeAlias=sag01
runtimeComponentId=IntegrationServer_default
```


9 Configuration Commands

■ sagcc get configuration common	210
■ sagcc get configuration compare	211
■ sagcc create configuration data	213
■ sagcc delete configuration data	216
■ sagcc get configuration data	218
■ sagcc update configuration data	221
■ sagcc get configuration instances	225
■ sagcc list configuration instances	227
■ sagcc get configuration types	229
■ sagcc list configuration types	230
■ sagcc exec configuration validation create	233
■ sagcc exec configuration validation delete	235
■ sagcc exec configuration validation update	237

sagcc get configuration common

Retrieves the schema for a specified configuration type.

Syntax

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

```
sagcc 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
<i>schema</i>	<p>Required. Specifies the schema you want to retrieve. You can only retrieve schemas for common configuration types.</p> <p>The following list the schema names you can specify:</p> <ul style="list-style-type: none"> ■ CommonSettings.xsd ■ EmailSettings.xsd ■ JDBCSettings.xsd ■ KeystoreTruststoreSettings.xsd ■ LicenseLocation.xsd ■ log4j.xsd ■ PortSettings.xsd
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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 [sagcc 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 Platform Manager 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”:

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

sagcc get configuration compare

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

Syntax

- Command Central syntax:

```
sagcc 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
<code>configurationTypeId=typeid</code>	<p>Required. Specifies the configuration type to compare.</p> <p>You can determine the IDs for configuration types using sagcc list configuration types.</p>
<code>runtimeComponentInfoId=id1</code> <code>runtimeComponentInfoId=id2</code> <code>[runtimeComponentInfoId=id3</code> <code>...</code> <code>runtimeComponentInfoId=idn]</code>	<p>Required. Specifies the information IDs of two or more run-time components that you want to compare.</p> <p>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 sagcc list landscape nodes. You can determine the IDs for run-time components using sagcc list inventory components.</p>
<code>[options]</code>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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
```

sagcc 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 an Integration 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:

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

■ Platform Manager syntax:

```
sagcc create configuration data componentid typeid
[sharedsecret=text_string]
[--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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to create a new instance of a configuration type.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>

Argument or Option	Description
<i>typeid</i>	<p>Required. Specifies the ID of the configuration type that identifies the type of instance you want to create.</p> <p>You can determine the IDs for configuration types using sagcc list configuration types.</p> <p>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.</p>
[sharedsecret= <i>text_string</i>]	<p>Optional. Specify a shared secret password that Platform Manager uses to secure any passwords in the configuration data.</p> <p>If you omit this parameter, Platform Manager uses the shared secret defined for the environment or for the whole landscape. For more information, see sagcc add security credentials sharedsecret.</p>
{--input -i} <i>file</i> {.xml .json .properties}	<p>Required. Identifies an input file that contains the configuration data. For more information, see "input" on page 177.</p> <div data-bbox="743 1213 1367 1894" style="background-color: #f0f0f0; padding: 10px;"> <p>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.</p> <p>Tip: To determine how to specify the data in the input file, use sagcc 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 sagcc get configuration data with the <code>--format xml</code> option to retrieve the data for an existing COMMON-PORTS instance in XML format.</p> </div>

Argument or Option	Description
<i>[options]</i>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

Usage Notes

- Not all run-time components support the `sagcc 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 `sagcc 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 `sagcc get configuration common` to retrieve schemas for common configuration types.

- The output from the `sagcc 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":

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

- The data to create a new instance of the COMMON-PROXY configuration type is in the `c:\inputs\proxy_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" and use "mysecret123" as the shared secret to secure the proxy configuration password:

```
sagcc create configuration data sag01 OSGI-SPM COMMON-PROXY
sharedsecret=mysecret123 --input
c:\inputs\proxy_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" on page 189](#) and ["username" on page 192](#). 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":

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

sagcc 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:

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

■ Platform Manager syntax:

```
sagcc 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
<code>node_alias</code>	Command Central only.

Argument or Option	Description
	<p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component from which you want to delete a configuration instance.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance you want to delete.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- You cannot delete some configuration data. In general, if you can create the configuration data, you can also usually delete it. The command returns an error if you attempt to perform an unsupported operation.
- You can use [sagcc 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":

```
sagcc 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" on page 189](#).

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”:

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

sagcc 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:

```
sagcc get configuration data node_alias componentid instanceid
[sharedsecret=text_string] [options]
```

- Platform Manager syntax:

```
sagcc get configuration data componentid instanceid
[sharedsecret=text_string] [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
<i>node_alias</i>	Command Central only. Required. Specifies the alias name of the installation in which the run-time component is installed.

Argument or Option	Description
	<p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve instance data.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance for which you want to retrieve data.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
[<i>sharedsecret=text_string</i>]	<p>Optional. Specify a shared secret password that Platform Manager uses to secure any passwords in the configuration data that you want to retrieve.</p> <p>If you omit this parameter, Platform Manager retrieves the configuration data using the shared secret defined for the environment or for the whole landscape. For more information, see sagcc add security credentials sharedsecret. If Platform Manager cannot find a shared secret that matches the shared secret specified when creating the configuration instance, Platform Manager does not include the passwords that the configuration contains in the output.</p> <p>When you omit this parameter:</p> <ul style="list-style-type: none"> ■ If the command is executed on Command Central, Platform Manager retrieves the configuration data using the shared secret defined for the environment or for the whole landscape. For more information, see sagcc add security credentials sharedsecret. ■ If the command is executed on Platform Manager, Platform Manager does not include the passwords that the configuration contains in the output.
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a</p>

Argument or Option	Description
	description of the options, see "Common Options" on page 167 .

Usage Notes

- Use [sagcc get configuration instances](#) or [sagcc 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:

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

- To retrieve instance data for the configuration instance with ID "COMMON-PROXY" that belongs to the run-time component with ID "OSGI-SPM" and runs in the installation with alias name "sag01", using "mysecret123" as the shared secret to secure the proxy configuration password, and save the configuration data with the encrypted password in the "configWithEncryptedData.xml" file:

```
sagcc get configuration data sag01 OSGI-SPM COMMON-PROXY
sharedsecret=mysecret123 --output
configWithEncryptedData.xml
```

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:

```
sagcc 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" on page 192](#) and ["password" on page 186](#).

sagcc 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:

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

■ Platform Manager syntax:

```
sagcc update configuration data componentid instanceid
[sharedsecret=text_string]
[--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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component to which the instance you want to update belongs.</p>

Argument or Option	Description
	<p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance for which you want to update data.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
<i>[sharedsecret=text_string]</i>	<p>Optional. Specify a shared secret password that Platform Manager uses to secure any passwords in the configuration data that you want to update.</p> <p>If you omit this parameter, Platform Manager uses the shared secret defined for the environment or for the whole landscape. For more information, see sagcc add security credentials sharedsecret.</p>
<pre>{--input -i} filename{.xml .json .properties}</pre>	<p>Required. Identifies an input file that contains the updated configuration data. For more information, see "input" on page 177.</p>
	<p>Note:Based on the type of configuration instance you are attempting to create, all file types (.xml, .json, and .properties) might not be supported.</p>
	<p>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 be supported when creating instances of a product-specific configuration type.</p>
	<p>Tip: To determine how to specify the data in the input file, use sagcc 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 sagcc get configuration data with the <code>--format xml</code> option to retrieve the data for</p>

Argument or Option	Description
	the COMMON-PORTS instance in XML format.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

Usage Notes

- The data in the input file must match the expected schema for the configuration type.
- You can use [sagcc 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 use [sagcc 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 use [sagcc get configuration common](#) to retrieve schemas for common configuration types.

- You can specify the content type for the output data for the command either in the Accept header of the REST client or by adding the `--output-format` option. For example, if you add `--output-format text`, the command displays data on the console in plain text format. For more information, see "[output-format](#)" on page 183.

If you do not specify the content type for the output data, the command returns the default output format, XML.

- When using an XML input data file to update a configuration instance, use the `sagcc get configuration data` command with the `--output | -o filename.xml` option to retrieve a copy of the XML input data file for the configuration instance you require.

After including the required updated data in the copy of the XML data file, you can use the `sagcc update configuration data` command to update the configuration instance.

Examples 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”:

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

- The `Software AG_directory/profiles/CCE/configuration/security/user.txt` file is updated with a new password that replaces the default Command Central administrator password. To update the modified user.txt file for the instance with ID “ENGINE SIN-INTERNAL-USERS-users.txt” and the run-time component with ID “OSGI-SPM”, which is installed in the installation with alias name “sag01”:

```
sagcc update configuration data sag01 OSGI-SPM-ENGINE
SIN-INTERNAL-USERS-users.txt --input
Software AG_directory
/profiles/CCE/configuration/security/users.txt
```

- To update instance data for the configuration instance with ID “COMMON-PROXY” that belongs to the run-time component with ID “OSGI-SPM” and runs in the installation with alias name “sag01”, using “mysecret123” as the shared secret to secure the proxy configuration password, and save the updated configuration data with the encrypted password in the “configWithEncryptedData.xml” file:

```
sagcc update configuration data sag01 OSGI-SPM COMMON-PROXY
sharedsecret=mysecret123 --input
configWithEncryptedData.xml
```

Because the `{--server | -s}` option is not specified, the command uses the default server. For more information, see ["server" on page 189](#).

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”:

```
sagcc 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" on page 192](#). The command specifies “secret” for the user’s password.

sagcc 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:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>

Argument or Option	Description
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve instance information.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance for which you want to retrieve information.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- To retrieve the data for a specific instance, use [sagcc 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:

```
sagcc 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:

```
sagcc 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" on page 192](#) and ["password" on page 186](#).

sagcc 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:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>

Argument or Option	Description
<i>componentid</i>	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 sagcc list inventory components .
[<i>instanceid</i>]	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 <i>componentid</i> argument.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- To retrieve the data for a specific instance, use [sagcc 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:

```
sagcc 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:

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

sagcc 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:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve a configuration type.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>

Argument or Option	Description
<i>typeid</i>	Required. Specifies the ID of the configuration type for which you want to retrieve information. You can determine the IDs for configuration types using sagcc list configuration types .
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- Run-time components support a set of configuration types. Use [sagcc 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:

```
sagcc 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:

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

sagcc 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:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to list configuration types.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[<i>typeid</i>]	<p>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-</p>

Argument or Option	Description
	time component identified by the <i>componentid</i> argument.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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 [sagcc 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:

```
sagcc 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:

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

sagcc exec configuration validation create

Validates the configuration instance data in the supplied input file. If the input data is valid, you can then use [sagcc create configuration data](#) to create a configuration instance.

Syntax

■ Command Central syntax:

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

■ Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>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.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>typeid</i>	<p>Required. Specifies the ID of the configuration type that identifies the type of instance data you want to validate.</p>

Argument or Option	Description
	<p>You can determine the IDs for configuration types using sagcc list configuration types.</p> <p>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.</p>
<pre>{--input -i} filename{.xml .json .properties}</pre>	<p>Required. Identifies an input file that contains the configuration data to validate. For more information, see "input" on page 177.</p> <div data-bbox="695 709 1370 1360" style="background-color: #f0f0f0; padding: 10px;"> <p>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.</p> <p>Tip: To determine how to specify the data in the input file, use sagcc 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 sagcc get configuration data with the <code>--format xml</code> option to retrieve the data for an existing COMMON-PORTS instance in XML format.</p> </div>
<pre>[options]</pre>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- Not all run-time components support the `sagcc 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 `sagcc create configuration data` command and supplying the validated input file.

- The `sagcc 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 `sagcc create configuration data` command to create a configuration instance.
 - When the command outputs error messages, the input data is not valid. The `sagcc 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":

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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":

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

sagcc exec configuration validation delete

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

Syntax

- Command Central syntax:

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

■ Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component that owns the instance.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance you want to check.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- Use this command to determine whether you can delete a configuration instance. This command does not delete the configuration instance.
- The `sagcc 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 [sagcc delete configuration data](#) command to delete the configuration instance.
- When the command outputs error messages, the check failed. The [sagcc 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”:

```
sagcc 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" on page 189](#).

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”:

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

sagcc 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 [sagcc delete configuration data](#) to update the configuration instance.

Syntax

- Command Central syntax:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>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.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance.</p> <p>You can determine the IDs for configuration instances using sagcc list configuration instances.</p>
{--input -i} <i>filename</i> {.xml .json .properties}	<p>Required. Identifies an input file that contains the configuration data to validate. For more information, see "input" on page 177.</p>

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 [sagcc 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 [sagcc get configuration data](#) with the `--format xml` option to retrieve

Argument or Option	Description
	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 "Common Options" on page 167 .

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 `sagcc delete configuration data` command and supplying the validated input file.
- The `sagcc 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 `sagcc delete configuration data` command to update the configuration instance.
 - When the command outputs error messages, the input data is not valid. The `sagcc 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":

```
sagcc 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" on page 189](#).

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":

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

10 Diagnostics Logs Commands

■ sagcc get diagnostics logs	242
■ sagcc get diagnostic logs export file	245
■ sagcc list diagnostics logs	247

sagcc 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:

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

■ To optionally specify search text:

```
sagcc 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:

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

■ To optionally specify search text:

```
sagcc 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
<i>node_alias</i>	Required. Only for Command Central. Specifies the alias name of the Platform Manager that

Argument or Option	Description
	<p>manages the run-time component for which you want to retrieve information.</p> <p>You can determine the alias name of the Platform Manager node by using sagcc list landscape nodes</p>
<i>runtime_componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve information.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<i>logid</i>	<p>Required. Specifies the ID of the log for which you want to retrieve information.</p> <p>You can determine the IDs for logs using sagcc list diagnostics logs.</p>
{full tail head}	<p>Required. Identifies the log entries you want to retrieve.</p> <ul style="list-style-type: none"> ■ Specify <code>full</code> to retrieve all log entries. ■ Specify <code>tail</code> to retrieve the most recent log entries from the end of the log file. ■ Specify <code>head</code> to retrieve entries from the beginning of the log file.
[lines= <i>number</i>]	<p>Optional. Use only with the <code>tail</code> or <code>head</code> parameters.</p> <p>Specifies the number of log entries to return. If you omit <code>lines=<i>number</i></code> the command returns 100 entries.</p>
[regex= <i>expression</i>]	<p>Optional. Narrows the retrieved log entries to those that meet the specified regular expression.</p>
[search= <i>text</i>]	<p>Optional. Narrows the retrieved log entries to those that contain the specified search text.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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](#).

- 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.

```
sagcc 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.

```
sagcc 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" on page 192](#). 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.

```
sagcc 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.

```
sagcc 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" on page 192](#). The command specifies “secret” for the user’s password.

sagcc 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:

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

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

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

- To export logs for all run-time components:

```
sagcc 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
<code>node_alias</code>	Required. Specifies the alias name of the Platform Manager that manages the run-time component for which you want to retrieve information.
<code>runtime_componentid</code>	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 sagcc list inventory components .
<code>[logid+logid...]</code>	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 sagcc list diagnostics logs .
<code>-o file</code>	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" on page 182 .
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

Examples When Executing on Command Central

- The run-time component with ID "OSGI-SPM" is managed by the Command Central 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".

```
sagcc 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 Command Central 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".

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

sagcc 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 [sagcc get diagnostics logs](#).

Syntax

- Command Central syntax:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	Required. Only for Command Central. Specifies the alias name of the Command Central that manages the run-time component for which you want to retrieve information.
<i>runtime_componentid</i>	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 sagcc list inventory components .

Argument or Option	Description
[<i>logid</i>]	Optional. Specifies the ID of the log for which you want to retrieve information.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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 Command Central 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":

```
sagcc 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 Command Central 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:

```
sagcc 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":

```
sagcc 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:

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

11 Instance Management Commands

■ sagcc create instances	250
■ sagcc delete instances	252
■ sagcc get instances	253
■ sagcc list instances supported products	254
■ sagcc update instances	255

sagcc create instances

Creates a new instance of an installed product.

Syntax

■ Command Central syntax:

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

■ Platform Manager syntax:

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

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which to create the product instance.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>product</i>	<p>Required. Specifies the product ID of the installed product or run-time component for which you want to create a new instance.</p> <p>Valid values for this option are only the product IDs included in the list of products returned from the <code>sagcc list instances <i>node_alias</i> supportedproducts</code> command.</p>
[<i>key=value</i>]	<p>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.</p>
[-i [<i>file</i> {.xml .json .properties}]	<p>Optional. Identifies an input file that contains the product instance data. For</p>

Argument or Option	Description
	<p>more information, see "input" on page 177.</p> <p>For the correct format of an XML properties file, see the Properties class in the Oracle Java Platform Standard Edition API specification.</p>
<code>[options]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- The command returns job information that you can monitor using the `sagcc list job manager` commands.
- When creating instances of Integration Server and My webMethods Server, you can use the `fixRepository=repoID` argument to ensure that all required fixes are applied on all product instances. For details about the argument, see the topic on creating product instances with the Command Central CLI in the administration guide of the product.

Example When Executing on Command Central

- To create the new instance for an installed Integration Server with instance name `"is-instance2"`, diagnostic port `"8083"`, JMX port `"10058"`, and primary port `"8081"` in the installation with alias name `"productionNode2"`:

```
sagcc create instances productionNode2 integrationServer
instance.name=is-instance2 diagnostic.port=8083
jmx.port=10058 primary.port=8081
```

Examples When Executing on Platform Manager

- To create the new instance for an installed Integration Server with instance name `"is-instance2"`, diagnostic port `"8083"`, JMX port `"10058"`, and primary port `"8081"`:

```
sagcc create instances integrationServer instance.name=is-instance2
diagnostic.port=8083 jmx.port=10058 primary.port=8081
```

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

```
sagcc 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:

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

sagcc delete instances

Deletes an existing instance of an installed product.

Syntax

- Command Central syntax:

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

- Platform Manager syntax:

```
sagcc delete instances componentid [options]
```

```
options :  
[--force]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	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 sagcc list landscape nodes .
<i>componentid</i>	Required. Specifies the ID of the run-time component that you want to delete.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

- The command returns job information that you can monitor using the `sagcc 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”:

```
sagcc delete instances sag01 integrationServer-default
```

Examples When Executing on Platform Manager

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

```
sagcc delete instances integrationServer-default
```

sagcc get instances

Retrieves a list of the configuration properties of a specified run-time component.

Syntax

- Command Central syntax:

```
sagcc get instances node_alias componentid [options]
```

- Platform Manager syntax:

```
sagcc get instances componentid [options]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	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 sagcc list landscape nodes .
<i>componentid</i>	Required. Specifies the ID of a run-time component for which you want to retrieve configuration data.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

To retrieve a list of configuration properties for a run-time component with ID “OSGI-IS” that runs in the installation with alias name “sag01”:

```
sagcc get instances sag01 OSGI-IS
```

Example When Executing on Platform Manager

To retrieve a list of configuration properties for a run-time component with name "OSGI-IS":

```
sagcc get instances OSGI-IS
```

sagcc list instances supported products

Retrieves a list of products that support instance management.

Syntax

- Command Central syntax:

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

- Platform Manager syntax:

```
sagcc list instances supportedproducts [options]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	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 sagcc list landscape nodes .
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

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

```
sagcc list instances sag01 supportedproducts
```

Example When Executing on Platform Manager

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

```
sagcc list instances supportedproducts
```

sagcc 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:

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

■ Platform Manager syntax:

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

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation in which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to update configuration properties.</p>
[<i>key=value</i>]	<p>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.</p>
[-i <i>file</i> {.xml .json .properties}]	<p>Optional. Identifies an input file that contains the new configuration data for the run-time component. For more information, see "input" on page 177.</p> <p>For the correct format of an XML properties file, see the Properties class in the Oracle Java Platform Standard Edition API specification.</p>

Argument or Option	Description
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

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”:

```
sagcc 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”:

```
sagcc 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:

```
sagcc 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:

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

12 Inventory Commands

■ sagcc exec inventory assets refresh	258
■ sagcc get inventory assets	258
■ sagcc list inventory assets	260
■ sagcc create inventory components attributes	262
■ sagcc get inventory components	264
■ sagcc get inventory components attributes	266
■ sagcc list inventory components	267
■ sagcc list inventory components attributes	273
■ sagcc update inventory components	274
■ sagcc update inventory components attributes	276
■ sagcc delete inventory components attributes	277
■ sagcc get inventory fixes	279
■ sagcc get inventory fixes compare	280
■ sagcc list inventory fixes	282
■ sagcc get inventory products	284
■ sagcc get inventory products compare	286
■ sagcc list inventory products	287

sagcc exec inventory assets refresh

Schedules an asynchronous update of the assets cache.

Important: The assets inventory commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

```
sagcc exec inventory assets refresh nodeAlias [options]
```

- Platform Manager syntax:

```
sagcc exec inventory assets refresh [options]
```

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation for which you want to refresh the assets cache.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
[<i>options</i>]	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Example When Executing on Command Central

To schedule an update of the assets cache on the local node:

```
sagcc exec inventory assets refresh local
```

sagcc get inventory assets

Retrieves information about all installed assets for all run-times on the specified node.

Important: The assets inventory commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

```
sagcc get inventory assets nodeAlias [refresh=true|false] [options]
```

- Platform Manager syntax:

```
sagcc get inventory assets [options]
```

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Command Central only. Required. Specifies the alias name of the installation for which you want to retrieve asset information. You can view a list of installations and their aliases using sagcc list landscape nodes .
[refresh=true false]	Optional. Specifies whether to schedule an asynchronous update of the assets cache. Valid values: <ul style="list-style-type: none"> ■ true - refresh the assets cache ■ false - do not refresh the assets cache Default: false
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To get information about the assets installed on the local installation:

```
sagcc get inventory assets local
```

sagcc list inventory assets

Lists assets that are either installed on the specified installation, or match the specified search criteria.

Important: The assets inventory commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

- To list all assets for all installations:

```
sagcc list inventory assets [options]
```

- To list assets installed on a specified installation:

```
sagcc list inventory assets nodeAlias
[options]
```

- To list assets that match specified search criteria:

```
sagcc list inventory assets [searchAttribute1=value1...] [start=from_number]
[size=count] [options]
```

- Platform Manager syntax:

```
sagcc list inventory assets [options]
```

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	<p>Command Central only.</p> <p>Specifies the alias name of the installation for which you want to retrieve asset information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>[searchAttribute1=value1...]</i>	<p>Command Central only.</p> <p>Optional. Narrows down the list of returned assets to only those that match the search criteria you specify. For information about the search criteria the command supports and how to specify them,</p>

Argument or Option	Description
	see the table in "Specifying Search Criteria for Inventory Commands" on page 271.
<code>[start=<i>from_number</i>]</code>	<p>Command Central only.</p> <p>Optional. Limits the results the command returns to those starting at the specified number in the results.</p> <p>For example, if you want to return information for the 5th through the 8th run-time assets in the results, use <code>start=5 size=4</code>.</p>
<code>[size=<i>count</i>]</code>	<p>Command Central only.</p> <p>Optional. Limits the number of results you want returned.</p> <p>For example, if you specify <code>size=1</code>, the command returns information for only one asset.</p>
<code>[options]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

If you do not specify an alias name, nor search criteria, the command lists information for all assets in all installations that Command Central manages.

Examples When Executing on Command Central

- To list all assets in the installation with alias name "local" and have the information returned to the output file "assets" in XML format:

```
sagcc list inventory assets local --format xml --output assets
```

- To restrict the number of returned assets to only 5:

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

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

```
sagcc list inventory assets start=10 size=6 --format xml
--password secret
```

- To list assets and use search criteria to narrow the results to only those that are installed in the installation with alias name “sag01” and return the output to the console in JavaScript Object Notation format:

```
sagcc list inventory assets nodeAlias=sag01
--format json --password secret
```

- To list assets 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:

```
sagcc list inventory assets logicalOperator=OR nodeAlias=sag01
nodeAlias=sag03 --format xml --password secret
```

- To list assets with name "Fibonacci" for the runtime of type "IS", installed on the "local" node:

```
sagcc list inventory assets nodeAlias=local name=Fibonacci runtimeType=IS
```

Example When Executing on Platform Manager

To list all assets on 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:

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

sagcc create inventory components attributes

Adds a new search attribute for a specified run-time component. The command supports only single-valued search attributes. A run-time component can have several search attributes, but each attribute takes a single value, for example:

```
attribute1=value1
attribute2=value2
attribute3=value3
...
```

Syntax

- Command Central syntax:

```
sagcc create inventory components attributes node_alias componentid
[attribute=value] [--input | -i} filename{.xml|.json}]
[options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>node_alias</code>	<p>Required. Specifies the alias name of the installation for which you want to add component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<code>componentid</code>	<p>Required. Specifies the ID of the run-time component for which you want to create a search attribute.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<code>[attribute=value]</code>	<p>Optional. The name of the new search attribute and its matching value.</p> <p>If the search attribute already exists, the command returns an error.</p>
<code>{[--input -i] filename{.xml .json}}</code>	<p>Optional. Identifies an input file that contains the data for the new search attribute.</p> <p>You retrieve attribute data for the <code>.xml json</code> file using the <code>sagcc get inventory components attributes</code> command:</p> <p>For more information, see "input" on page 177.</p>
<code>[options]</code>	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

After creating a new search attribute for a run-time component, you can include the new attribute as search criteria in a lifecycle or search command to execute an operation against all run-time components that match the new search attribute. For example, the `sagcc execute lifecycle start group=AB` command starts all run-time components included in the "AB" group.

For information about including search criteria in lifecycle commands, see ["Specifying Search Criteria for Lifecycle Commands" on page 359](#). For information about

including search criteria in search inventory commands, see ["Specifying Search Criteria for Inventory Commands" on page 271](#).

Examples When Executing on Command Central

- To create two search attributes, one with name "group" that matches the value "Test" and another with name "tenant" that matches the value "abc.com", for the run-time component that has the component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

```
sagcc create inventory components attributes
sag01 OSGI-SPM group=Test tenantId=abc.com --password secret
```

- To create a new search attribute, using the attribute data from the attributes.xml file, for the run-time component that has the component ID "OSGI-CCE" and is installed on the installation with the alias name "sag01":

```
sagcc create inventory components attributes
sag01 OSGI-CCE -i c:\inputs\attributes.xml --password secret
```

To retrieve attribute data for the attribute.xml file use the following command:

```
sagcc get inventory components attributes sag01 OSGI-CCE -f xml
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see ["username" on page 192](#). The command specifies "secret" for the user's password.

sagcc 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 a run-time component that functions on its own. These are referred to as *instances* in the Web user interface.
 - ENGINE for a run-time component that cannot function on its 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:

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

- Platform Manager syntax:

```
sagcc 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
<i>node_alias</i>	<p>Command Central only.</p> <p>Required. Specifies the alias name of the installation for which you want to retrieve component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve information.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

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

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see ["username" on page 192](#). 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:

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

sagcc get inventory components attributes

Retrieves the attribute value matching a search attribute for a run-time component. To retrieve data for all search attributes for a run-time component, use the [sagcc list inventory components attributes](#) command.

Syntax

- Command Central syntax:

```
sagcc get inventory components attributes node_alias componentid
[attribute] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	<p>Required. Specifies the alias name of the installation for which you want to retrieve component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to retrieve search attribute data.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[<i>attribute</i>]	<p>Optional. The name of the search attribute whose matching value you want to retrieve.</p> <p>When you do not specify a value for the attribute argument, the command lists all attributes for a run-time component and their matching values.</p>

Argument or Option	Description
[<i>options</i>]	Optional. The command supports all options supported by Command Central. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

- To retrieve the value for a search attribute with name "group" for the run-time component that has component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

- To include the name "group" and value "AB" of the search attribute with headers in the output:

```
sagcc get inventory components attributes sag01 OSGI-SPM
group -p secret
```

Output:

Name	Value
group	AB

- To include only the value of the "group" attribute, for example "AB", without any headers in the output:

```
sagcc get inventory components attributes sag01 OSGI-SPM
group properties=value includeHeaders=false -p secret
```

- To retrieve all search attributes and their matching values for the run-time component that has component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01", and has two search attributes "group" and "tenantId":

```
sagcc get inventory components attributes sag01 OSGI-SPM -p manage
```

Output:

Name	Value
group	AB
tenantId	1234

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "[username](#)" on page 192. The command specifies "secret" for the user's password.

sagcc 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:

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

- To list components that match specified search criteria:

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

- Platform Manager syntax:

```
sagcc 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]	<p>Command Central only.</p> <p>Optional. Specifies the alias name of the installation for which you want to retrieve component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p> <p>If you do not specify an alias name nor search criteria, the command lists information for all run-time components for all installations that Command Central manages.</p>

Argument or Option	Description
[<i>componentid</i>]	<p>Optional. Specifies the ID of the run-time component for which you want to retrieve information.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[<i>criteria</i>]	<p>Command Central only.</p> <p>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 271.</p>
[<i>start=number</i>]	<p>Command Central only.</p> <p>Optional. Limits the results the command returns those starting at specified number in the results.</p> <p>For example, if you want to return information for the 5th through the 8th run-time components in the results, use <code>start=5 size=4</code>.</p>
[<i>size=number</i>]	<p>Command Central only.</p> <p>Optional. Limits the number of results you want returned.</p> <p>For example, if you specify <code>size=1</code>, the command returns information for only one run-time component.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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:

```
sagcc 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

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

Specifying Search Criteria for Inventory Commands

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

```
attribute1=value1 attribute2=value2 ...
```

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

Supported Search Criteria for Inventory Commands

Command	Attribute Names
<code>sagcc list inventory assets</code>	<ul style="list-style-type: none"> ■ <code>nodeAlias</code> The alias of the installation on which the asset is installed ■ <code>runtimeComponentId</code> The ID of the run-time component to which the asset belongs. ■ <code>name</code> The name of the installed asset. ■ <code>version</code> The version of the installed asset. ■ <code>runtimeType</code> The type of the asset runtime. Valid values to match: <ul style="list-style-type: none"> <code>AppPlatform</code> <code>BPM</code> <code>DES</code> <code>EDA</code> <code>IS</code> <code>MWS</code> <code>RULES</code> <code>RULESMWS</code> <code>TN</code> ■ <code>type</code> The type of the asset. The type is product-specific, for example

Command	Attribute Names
	<p>for Integration Server the type is <code>ispackage</code>.</p> <p>For information about the asset types, see <i>webMethods Deployer User's Guide</i>.</p>
<pre>sagcc list inventory components</pre>	<ul style="list-style-type: none"> ■ nodeName ■ nodeAlias ■ nodeUrl ■ environmentName ■ environmentAlias ■ runtimeComponentInfoId ■ runtimeComponentId ■ runtimeComponentDisplayName ■ runtimeComponentProductId ■ runtimeComponentCategory ■ runtimeComponentRuntimeStatus ■ runtimeComponentRuntimeParentId <p>You can combine any of the pre-defined attribute names in this list with new search attributes added using sagcc create inventory components attributes.</p>
<pre>sagcc list inventory fixes</pre>	<ul style="list-style-type: none"> ■ nodeName ■ nodeAlias ■ nodeUrl ■ environmentName ■ environmentAlias ■ fixId ■ fixDisplayName ■ fixVersion ■ fixGroup ■ fixProducts
<pre>sagcc list inventory products</pre>	<ul style="list-style-type: none"> ■ nodeName ■ nodeAlias ■ nodeUrl ■ environmentName ■ environmentAlias ■ productId ■ productCanonicalId ■ productDisplayName ■ productParentId ■ productGroup

Command	Attribute Names
	<ul style="list-style-type: none"> ■ productProfileDir ■ productCode ■ productVersion ■ productInstallTime

Specifying the Value

When specifying the value, you can include the * pattern-matching character to match multiple characters or the ? character to match a single character. 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.

You can also use the * and ? pattern-matching characters to search for attribute names. For example, if you want to list the search attributes attribute1=value1 and attribute2=value2 for different run-time components, use the following search criterion:

```
attribute?=value?
```

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 attributes. 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
```

sagcc list inventory components attributes

Lists all search attributes for a run-time component.

Syntax

- Command Central syntax:

```
sagcc list inventory components attributes node_alias componentid [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>node_alias</code>	<p>Required. Specifies the alias name of the installation for which you want to list component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<code>componentid</code>	<p>Required. Specifies the ID of the run-time component for which you want to list the existing search attributes.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
<code>[options]</code>	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see "Common Options" on page 167.</p>

Examples When Executing on Platform Manager

- To list all search attributes for the run-time component that has the component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

```
sagcc list inventory components attributes sag01 OSGI-SPM --password secret
```

- To list all search attributes for the run-time component that has the component ID "OSGI-CCE" and is installed on the installation with the alias name "sag01":

```
sagcc list inventory components attributes sag01 OSGI-CCE
-f xml -o attributes.xml --password secret
```

The command writes the output to the attributes.xml file. You can create or update search attributes for a run-time component in the attribute.xml file using the create or update inventory attributes commands, For example, to update the attributes in the attributes.xml file:

```
sagcc update inventory components attributes node_alias componentid -i attributes.xml
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see ["username" on page 192](#). The command specifies "secret" for the user's password.

sagcc update inventory components

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

Syntax

■ Command Central syntax:

```
sagcc 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
<i>node_alias</i>	<p>Required. Specifies the alias name of the installation for which you want to retrieve component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component that you want to update.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
{--input -i} <i>filename</i> {.xml .json .properties}	<p>Required. Identifies an input file that contains the updated data for the run-time component. For more information, see "input" on page 177.</p>
[<i>options</i>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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 [sagcc 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:

```
sagcc 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" on page 189](#), ["username" on page 192](#), and ["password" on page 186](#).

sagcc update inventory components attributes

Updates the value that matches an existing search attribute for a run-time component.

Syntax

- Command Central syntax:

```
sagcc update inventory components attributes node_alias componentid
[attribute=value] [--input | -i] filename{.xml|.json} [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	<p>Required. Specifies the alias name of the installation for which you want to update component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to update the value of a search attribute.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[attribute=value]	<p>Optional. The name of the search attribute and the new matching value for the property.</p>

Argument or Option	Description
<code>[{--input -i} filename{.xml .json}]</code>	<p>Optional. Identifies an input file that contains the update data for the search attribute.</p> <p>You retrieve attribute data for the <code>.xml json</code> file using the <code>c sagcc get inventory components attributes</code> command.</p> <p>For more information, see "input" on page 177.</p>
<code>[options]</code>	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

You can execute the command either with the `[attribute=value]`, or the `[{--input | -i} filename{.xml|.json}]` argument, but not with both arguments in the same command.

Examples When Executing on Command Central

- To change the existing value of a search property with name "group" to the new value "Production" for the run-time component that has the component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

```
sagcc update inventory components attributes sag01 OSGI-SPM
group=Production --password secret
```

- To update the value of a search property, using the property data from the `attributes.xml` file, for the run-time component that has the component ID "OSGI-CCE" and is installed on the installation with the alias name "sag01":

```
sagcc update inventory components attributes sag01 OSGI-CCE
-i c:\inputs\attributes.xml --password secret
```

- To update a search property with name "group", using the property data from the `attributes.xml` file, for the run-time component that has the component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

```
sagcc update inventory components attributes sag01 OSGI-SPM
group -i c:\inputs\attributes.xml --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see ["username" on page 192](#). The command specifies "secret" for the user's password.

sagcc delete inventory components attributes

Deletes an existing search attribute for a run-time component.

Syntax

- Command Central syntax:

```
sagcc delete inventory components attributes node_alias componentid
[attribute] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	<p>Required. Specifies the alias name of the installation for which you want to delete component information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to delete a search attribute.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[attribute]	<p>Optional. The name of the search attribute that you want to delete.</p>
[options]	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

If you do not specify a search attribute name, the command deletes all existing search attributes for a run-time component.

Examples When Executing on Command Central

- To delete a search attribute with name "group" for the run-time component that has the component ID "OSGI-SPM" and is installed on the installation with the alias name "sag01":

```
sagcc delete inventory components attributes sag01
OSGI-SPM group --password secret
```

- To delete all search attributes for the run-time component that has the component ID “OSGI-CCE” and is installed on the installation with the alias name “sag01”:

```
sagcc delete inventory components attributes sag01
OSGI-CCE --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see ["username" on page 192](#). The command specifies “secret” for the user’s password.

sagcc get inventory fixes

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

- Fix ID
- Fix name
- Version of the fix
- Product to which the fix is applied
- The date and time when the fix was installed

Syntax

- Command Central syntax:

```
sagcc get inventory fixes nodeAlias fixId [options]
```

- Platform Manager syntax:

```
sagcc get 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
[--password | -p] password
[--quiet | -q]
[--server | -s] url
[--username | -u] user_name
```

Arguments and Options

Argument or Option	Description
[<i>nodeAlias</i>]	Command Central only. Required. Specifies the alias name of the installation for which you want to retrieve fix information. If you do not specify an alias name, the command lists all fixes in all installations that Command Central manages.

Argument or Option	Description
	You can view a list of installations and their aliases using <code>sagcc list landscape nodes</code> .
<code>fixId</code>	Required. Specifies the fix ID of the fix for which you want to retrieve product information. You can determine the IDs for fixes using " <code>sagcc list inventory fixes</code> " on page 282.
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " <code>Common Options</code> " on page 167.

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 fix that has the ID "wMFix.CCE.Fix1" and is installed on the installation with the alias name "sag01", and have the output returned to the console in JavaScript Object Notation format:

```
sagcc get inventory fixes sag01 wMFix.CCE.Fix1 --server http://rubicon:8090/cee --format json --password secret
```

Because the `{--username | -u}` option is not specified, the command uses the default user name. For more information, see "`username`" on page 192. The command specifies "secret" for the user's password.

Example When Executing on Platform Manager

To retrieve information for the fix that has the ID "wMFix.CCE.Fix1" 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:

```
sagcc get inventory products wMFix.CCE.Fix1 --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`" on page 192. The command specifies "secret" for the user's password.

sagcc get inventory fixes compare

Compares the fixes installed in two or more installations.

Syntax

- Command Central syntax:

```
sagcc 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 sagcc 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 "Common Options" on page 167 .

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:

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

sagcc 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
- The date and time when the fix was installed

Syntax

- Command Central syntax:

- To list information for a specified installation:

```
sagcc list inventory fixes [nodeAlias] [options]
```

- To list fixes that match specified search criteria:

```
sagcc list inventory fixes [searchCriteria] [start=from_number]
[size=count] [options]
```

- Platform Manager syntax:

```
sagcc list inventory fixes [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
[<i>nodeAlias</i>]	Command Central only. Optional. Specifies the alias name of the installation for which you want to retrieve fix information. You can view a list of installations and their aliases using sagcc list landscape nodes .

Argument or Option	Description
	If you do not specify an alias name nor search criteria, the command lists all fixes in all installations that Command Central manages.
[<i>searchCriteria</i>]	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 271.
[<i>start=from_number</i>]	Command Central only. Optional. Limits the results the command returns to those starting at the specified number in the results. For example, if you want to return information for the 5th through the 8th products in the results, use <code>start=5 size=4</code> .
[<i>size=count</i>]	Command Central only. Optional. Limits the number of results you want returned. For example, if you specify <code>size=1</code> , the command returns information for only one product.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.

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 installation with alias "sag01" and have the output returned to the console in JavaScript Object Notation format:

```
sagcc list inventory fixes sag01 --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"](#) on page 192. 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:

```
sagcc 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:

```
sagcc list inventory fixes sag01 size=5 --format xml --output productlist
--server http://rubicon:8090/cce --username Administrator
--password manage
```

- To list the fixes that are version 9.12 or later and also contain “wMFix” in their fix IDs, and return the output to the console in JavaScript Object Notation format:

```
sagcc list inventory fixes fixversion=9.12* 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" on page 189](#) and ["username" on page 192](#). 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

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

sagcc 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:

```
sagcc get inventory products nodeAlias productId [options]
```

- Platform Manager syntax:

```
sagcc 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
<i>[nodeAlias]</i>	<p>Command Central only.</p> <p>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.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<i>productId</i>	<p>Required. Specifies the product ID of the product for which you want to retrieve product information.</p> <p>You can determine the IDs for products using sagcc list inventory products.</p>
<i>[options]</i>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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" on page 192](#). 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:

```
sagcc 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" on page 192](#). The command specifies “secret” for the user’s password.

sagcc get inventory products compare

Compares the products installed in two or more installations.

Syntax

■ Command Central syntax:

```
sagcc 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
<code>nodeAlias=alias1</code> <code>nodeAlias=alias2</code> <code>[nodeAlias=alias3</code> <code>...</code> <code>nodeAlias=aliasn]</code>	<p>Required. Specifies the alias names of two or more installations that you want to compare.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<code>[options]</code>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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" on page 192](#). The command specifies “secret” for the user’s password.

sagcc 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:

```
sagcc list inventory products [nodeAlias] [options]
```

- To list products that match specified search criteria:

```
sagcc list inventory products [searchCriteria] [start=from_number]
[size=count] [options]
```

- Platform Manager syntax:

```
sagcc list inventory products [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
[<i>nodeAlias</i>]	<p>Command Central only.</p> <p>Optional. Specifies the alias name of the installation for which you want to retrieve product information.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p> <p>If you do not specify an alias name nor search criteria, the command lists all products in all installations that Command Central manages.</p>
[<i>searchCriteria</i>]	<p>Command Central only.</p> <p>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 271.</p>
[<i>start=from_number</i>]	<p>Command Central only.</p> <p>Optional. Limits the results the command returns to those starting at the specified number in the results.</p>

Argument or Option	Description
	For example, if you want to return information for the 5th through the 8th products in the results, use <code>start=5 size=4</code> .
<code>[size=count]</code>	<p>Command Central only.</p> <p>Optional. Limits the number of results you want returned.</p> <p>For example, if you specify <code>size=1</code>, the command returns information for only one product.</p>
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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:

```
sagcc 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:

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

- To list products that are version 9.12 or later and also contain "Platform" in their display name and return the output to the console in JavaScript Object Notation format:

```
sagcc list inventory products productVersion=9.12*
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" on page 189](#) and ["username" on page 192](#). 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

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

13 Jobmanager Jobs Commands

■ sagcc delete jobmanager jobs	292
■ sagcc delete jobmanager landscapejobs	293
■ sagcc list jobmanager jobs	294
■ sagcc list jobmanager landscapejobs	295

sagcc delete jobmanager jobs

Attempts to cancel a long-running job. The success of the cancel operation depends on the status of the job as described in the Usage Notes section for this command.

Syntax

- Command Central syntax:

```
sagcc delete jobmanager jobs jobid [options]
```

- Platform Manager syntax:

```
sagcc delete jobmanager jobs jobid [options]
```

```
options :  
[--force]
```

Arguments and Options

Argument or Option	Description
<i>jobid</i>	Required. Specifies the ID of the job that you want to cancel.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

- When the command cancels a job successfully, it returns code 200 OK. If the command returns code 202 Accepted, the job might not be canceled. You should verify the status of the job using the `sagcc get jobmanager jobs` command.
- If the status of a long-running job is RUNNING, the command attempts to cancel the job, but might fail to cancel it.
- When the status of a long-running job is DONE, CANCELED, SCHEDULED, TIMEDOUT, or ERROR, the command removes the job from the jobs list.

Example When Executing on Command Central

To cancel a long-running job with ID "2" that is running in the installation that the Command Central server with host name "rubicon" and port "8090" manages:

```
sagcc delete jobmanager jobs 2 --server http://rubicon:8090/cce
```

Example When Executing on Platform Manager

To cancel a long-running job with ID “3” that is running in the installation that the Platform Manager server with host name “rubicon2” and port “8092” manages:

```
sagcc delete jobmanager jobs 3 --server http://rubicon2:8092/spm
```

sagcc delete jobmanager landscapejobs

Attempts to cancel a long-running job on any of the Platform Managers that are managed by the same Command Central instance. The success of the cancel operation depends on the status of the job as described in the Usage Notes section for this command.

Syntax

- Command Central syntax:

```
sagcc delete jobmanager landscapejobs node_alias jobid [options]
```

- Not supported on Platform Manager.

```
options :  
[--force]
```

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	Specifies the alias name of the installation for which you want to cancel a long-running job.
<i>jobid</i>	Required. Specifies the ID of the job that you want to cancel.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- When the command cancels a job successfully, it returns code 200 OK. If the command returns code 202 Accepted, the job might not be canceled. You should verify the status of the job using the `sagcc get jobmanager jobs` command.
- If the status of a long-running job is RUNNING, the command attempts to cancel the job, but might fail to cancel it.

- When the status of a long-running job is DONE, CANCELED, SCHEDULED, TIMEDOUT, or ERROR, the command removes the job from the jobs list.

Example When Executed on Command Central

To cancel the job with ID “5” that is running in the installation with alias “prod-is”:

```
sagcc delete jobmanager landscapejobs prod-is 5
```

sagcc 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 `sagcc exec lifecycle` command might take several seconds to complete.

Syntax

- Command Central syntax:

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

- Platform Manager syntax:

```
sagcc 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
[<i>jobid</i>]	Optional. Specifies the ID of the job for which you want to retrieve information.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

Usage Notes

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

Example When Executing on Command Central

To retrieve information for the job with ID “2” that is running in the installation that the Command Central server with host name “rubicon” and port “8090” manages, and have the output returned to the console in XML format:

```
sagcc get jobmanager jobs 2 --server http://rubicon:8090/cce --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" on page 192](#) and ["password" on page 186](#).

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 “secret”:

```
sagcc list jobmanager jobs --server http://rubicon2:8092/spm
--username Administrator --password secret
```

- To retrieve information for the job with ID “3” 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:

```
sagcc 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" on page 192](#) and ["password" on page 186](#).

sagcc list jobmanager landscapejobs

Lists information about long-running jobs in the landscape. The command lists all jobs running on any of the Platform Managers managed by the same Command Central instance.

Syntax

- Command Central syntax:

```
sagcc list jobmanager landscapejobs node_alias
[jobId] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	Specifies the alias name of the installation for which you want to retrieve job details.
[<i>jobid</i>]	Optional. Specifies the ID of the job for which you want to retrieve information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

If you omit [*jobid*], the command retrieves the information for all long-running jobs on the Platform Manager installation specified in the *node_alias* argument.

Examples when Executing on Command Central

- To retrieve information for all the long-running jobs in the installation with alias "prod-is", using the authorization of the user with user name "Administrator" and password "secret":

```
sagcc list jobmanager landscapejobs prod-is
--username Administrator --password secret
```

- To retrieve information for the job with ID "5" that is running in the installation with alias "prod-is":

```
sagcc list jobmanager landscapejobs prod-is 5
```

14 Landscape Commands

■ sagcc add landscape environments nodes	298
■ sagcc create landscape environments	299
■ sagcc delete landscape environments	301
■ sagcc get landscape environments	302
■ sagcc list landscape environments	303
■ sagcc remove landscape environments nodes	304
■ sagcc update landscape environments	306
■ sagcc create landscape nodes	308
■ sagcc delete landscape nodes	310
■ sagcc exec landscape nodes generateNodeId	311
■ sagcc get landscape nodes	313
■ sagcc list landscape nodes	314
■ sagcc update landscape nodes	315

sagcc add landscape environments nodes

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

Syntax

■ Command Central syntax:

```
sagcc 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
<i>env_alias</i>	Required. Specifies the alias name of the environment to which you want to add one or more installations.
<i>nodes</i>	Required. Specifies a required keyword indicating you are adding installations (also known as <i>nodes</i>) to an environment.
nodeAlias= <i>alias1</i> [nodeAlias= <i>alias2</i> ... nodeAlias= <i>aliasn</i>]	Required. Specifies the alias name(s) of one or more installation(s) that you want to add to the environment.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

Usage Notes

- Use the [sagcc create landscape environments](#) command to create an environment.

- Use the `sagcc 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" on page 189](#), ["username" on page 192](#), and ["password" on page 186](#).

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

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

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

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

sagcc 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:

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

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

```
sagcc 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
<code>alias=<i>env_alias</i></code>	<p>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.</p> <p>Valid characters are ASCII characters, numbers, hyphen (-), underscore (_), and period (.). Spaces are not allowed.</p>
<code>[name=<i>name</i>]</code>	<p>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:</p> <pre>name="Dev Environment"</pre> <p>If you do not specify a display name, the command uses the value you supply for the environment alias name.</p>
<code>[description=<i>description</i>]</code>	<p>Optional. Specifies a description for the new environment. If you use a value that includes spaces, place quotes around the value, for example:</p> <pre>description="A description with spaces"</pre>
<code>[{--input -i} <i>filename</i>{.xml .json}]</code>	<p>Optional. Identifies an input file that contains the data for the new environment. For more information, see "input" on page 177.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Tip: To determine how to specify the data in the input file, use sagcc get landscape environments to retrieve data for an existing environment. For example, if you want to use an XML input file, use sagcc get landscape environments with the <code>--format xml</code> option to retrieve the data in XML format.</p> </div>
<code>[<i>options</i>]</code>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- After creating the new environment, use `sagcc 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”:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). The command specifies “secret” for the user’s password

sagcc delete landscape environments

Deletes a specified environment.

Syntax

- Command Central syntax:

```
sagcc 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
<code>env_alias</code>	Optional. Specifies the alias name of the environment you want to delete.
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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 [sagcc remove landscape environments nodes](#) command.
- If you want to remove an installation from Command Central management, use the [sagcc 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”:

```
sagcc 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" on page 189](#).

sagcc 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:

```
sagcc 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
<code>env_alias</code>	Required. Specifies the alias name of the environment whose information you want to retrieve.
[<code>nodes</code>]	Optional. Indicates you want the command to return the information about the installations in the environment. If you omit the <code>nodes</code> parameter, the returned information will not include the list of installations in the environment.
[<code>options</code>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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:

```
sagcc 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:

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

sagcc 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:

```
sagcc 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
[<i>env_alias</i>]	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.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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:

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

sagcc remove landscape environments nodes

Removes one or more installations from a specified environment.

Syntax

- Command Central syntax:

```
sagcc 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
<i>env_alias</i>	Required. Specifies the alias name of the environment from which you want to remove one or more installations.
<i>nodeAlias=alias1</i> [<i>nodeAlias=alias2</i> ... <i>nodeAlias=aliasn</i>]	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.
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

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 [sagcc 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”:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). The command specifies “secret” for the user’s password

sagcc 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:

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

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

```
sagcc 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
<i>env_alias</i>	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.

Argument or Option	Description
	<p>If you use a value that includes spaces, place quotes around the value, for example:</p> <pre>name="Dev Environment"</pre>
<code>[description=<i>description</i>]</code>	<p>Optional. Specifies the updated description for the environment.</p> <p>If you use a value that includes spaces, place quotes around the value, for example:</p> <pre>description="A description with spaces"</pre>
<code>[{--input -i} filename{.xml .json}]</code>	<p>Optional. Identifies an input file that contains the updated data for the environment. For more information, see "input" on page 177.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Tip: To determine how to specify the data in the input file, use sagcc 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 sagcc get landscape environments with the <code>--format xml</code> option to retrieve the data in XML format.</p> </div>
<code>[options]</code>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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" on](#)

page 189 and "username" on page 192. The command specifies "secret" for the user's password.

sagcc 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:

```
sagcc 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:

```
sagcc 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
<code>alias=node_alias</code>	<p>Required. Specifies the alias name you want to assign to the installation. The name must be unique among all installations that Command Central manages.</p> <p>Valid characters are ASCII characters, numbers, hyphen (-), underscore (_), and period (.). Spaces are not allowed.</p>
<code>url=url</code>	<p>Required. Specifies the URL of the Command Central that manages the installation. For example:</p> <pre>http://rubicon:8092</pre>

Argument or Option	Description
	<p>When specifying the URL, if you omit the port number, the command uses "8092", which is the default port for a Command Central server.</p>
<p>[name=<i>name</i>]</p>	<p>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:</p> <pre data-bbox="704 625 1370 653">name="my installation"</pre> <p>If you do not specify a display name, the command uses the value you specify for the alias name.</p>
<p>[description=<i>description</i>]</p>	<p>Optional. Specifies a description for the installation. If you use a value that includes spaces, place quotes around the value, for example:</p> <pre data-bbox="704 968 1370 995">description="A description with spaces"</pre>
<p>[{--input -i} filename{.xml .json}]</p>	<p>Optional. Identifies an input file that contains the data for the new node. For more information, see "input" on page 177.</p> <div data-bbox="704 1146 1370 1415" style="background-color: #f0f0f0; padding: 10px;"> <p>Tip: To determine how to specify the data in the input file, use sagcc get landscape nodes to retrieve data for an existing node. For example, if you want to use an XML input file, use sagcc get landscape nodes with the <code>--format xml</code> option to retrieve the data in XML format.</p> </div>
<p>[<i>options</i>]</p>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- Use the `sagcc create landscape nodes` command to create an installation that is not associated with an environment. After creating the installation, you can use the `sagcc 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”:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). The command specifies “secret” for the user’s password

sagcc delete landscape nodes

Removes an installation from being centrally managed via Command Central.

Syntax

- Command Central syntax:

```
sagcc delete landscape nodes [{alias | nodeAlias=alias1
[nodeAlias=alias2 ... nodeAlias=aliasn]}] [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
[<code>{alias nodeAlias=alias1</code> [<code>nodeAlias=alias2 ...</code> <code>nodeAlias=aliasn</code>]	Optional. Specifies the alias name(s) of the installation(s) you want to remove. If you execute the <code>sagcc delete landscape nodes</code> 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: <pre>sagcc delete landscape nodes alias</pre>

Argument or Option	Description
	To remove multiple installations, supply each name using the <code>nodeAlias=alias</code> format to identify each node to remove.
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- The `sagcc 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 `sagcc 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" on page 189](#), ["username" on page 192](#), and ["password" on page 186](#).

- To remove the installation with alias "mws01":

```
sagcc delete landscape nodes mws01
```

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

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

- To remove all installations:

```
sagcc delete landscape nodes
```

sagcc 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:

```
sagcc 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
<code>node_alias</code>	<p>Required. Specifies the alias name of the installation for which you want to generate an ID.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
<code>[options]</code>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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 [sagcc 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 `sagcc 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":

```
sagcc 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" on page 189](#).

sagcc 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 Command Central that manages the installation
- Status of the Command Central that manages the installation

Syntax

- Command Central syntax:

```
sagcc 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
<i>node_alias</i>	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 sagcc list landscape nodes .
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

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:

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

sagcc 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:

```
sagcc 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
<code>[node_alias]</code>	Optional. Specifies the alias name of the installation for which you want to list information.
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

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 Command Central 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:

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

sagcc 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:

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

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

```
sagcc 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
<i>node_alias</i>	<p>Required. Specifies the alias name of the installation whose description you want to update.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
[name= <i>name</i>]	<p>Optional. Specifies the updated display name for the installation.</p> <p>If you use a value that includes spaces, place quotes around the value, for example:</p> <pre>name="My installation"</pre>
[description= <i>description</i>]	<p>Optional. Specifies the updated description for the installation.</p> <p>If you use a value that includes spaces, place quotes around the value, for example:</p> <pre>description="A description with spaces"</pre>
[/--input -i} <i>filename</i> {.xml .json}]	<p>Optional. Identifies an input file that contains the updated data for the landscape. For more information, see "input" on page 177.</p>

Argument or Option	Description
	<p>Tip: To determine how to specify the data in the input file, use the sagcc 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 sagcc get landscape nodes with the <code>--format xml</code> option to retrieve the data in XML format.</p>
<p><code>[options]</code></p>	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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":

```
sagcc 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](#)" on page 189 and "[username](#)" on page 192. The command specifies "secret" for the user's password.

15 License Keys Commands

■ sagcc add license-tools keys	320
■ sagcc get license-tools keys	321
■ sagcc list license-tools keys	322
■ sagcc update license-tools keys	325
■ sagcc delete license-tools keys	326
■ sagcc update configuration license	326

sagcc add license-tools keys

Adds a product license key file with the specified alias to the Command Central license key manager.

The license key manager enables you to manage Software AG licenses keys and use them for template-based provisioning.

Syntax

- Command Central syntax:

```
sagcc add license-tools keys licenseKeyAlias
{--input|-i} filename [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>licenseKeyAlias</i>	Required. The alias of the license key file to add. Specify an alias that is unique across all license keys in the license key manager. Valid characters are ASCII characters, numbers, hyphen (-), underscore (_), and period (.). Spaces are not allowed.
{--input -i} <i>filename</i>	Required. Specifies the name of the input file. For more information, see "input" on page 177 .
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

The license key manager supports two types of license keys:

- **Standard.** License keys for Software AG products that are produced and managed using standard Software AG licensing tools and processes. The standard keys are in .xml format.
- **Custom.** License keys for Software AG products that have not yet adopted the standard Software AG licensing tools and processes, for example Terracotta. The custom keys are in any format that a product currently supports, for example a text file.

Examples When Executing on Command Central

- To add the license key file “Integration_Server96WinDesktop.xml” with the alias “PIE96WinDesktop” to the license key manager:

```
sagcc add license-tools keys PIE96WinDesktop
-i c:\Licenses\Integration_Server96WinDesktop.xml
```

- To add the license key file “terracotta-license.key” with alias “Terracotta” to the license key repository:

```
sagcc add license-tools keys Terracotta
-i c:\Licenses\terracotta-license.key
```

sagcc get license-tools keys

Retrieves a license key with the specified alias from the license key manager and displays the contents of the license key file.

Syntax

- Command Central syntax:

```
sagcc get license-tools keys licenseKeyAlias
[{-o|--output} filename] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>licenseKeyAlias</i>	Required. The alias of the license key file to retrieve.
{-o --output <i>filename</i> }	Optional. Specifies the name of the output file. For more information, see "output" on page 182 .
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

The license key manager supports two types of license keys:

- **Standard.** License keys for Software AG products that are produced and managed using standard Software AG licensing tools and processes. The standard keys are in .xml format.
- **Custom.** License keys for Software AG products that have not yet adopted the standard Software AG licensing tools and processes, for example Terracotta. The

custom keys are in any format that a product currently supports, for example a text file.

Examples When Executing on Command Central

To retrieve the contents of a license key file with the alias "PIE96WinDesktop":

```
sagcc get license-tools keys PIE96WinDesktop
-o c:\temp\PIE96WinDesktop.xml
```

sagcc list license-tools keys

Returns all license keys available in the license key manager or a list of license keys based on a set of filters. The filters include:

- Product code
- Product ID
- Operating system
- Platform
- Installation type
- License key type

Syntax

- Command Central syntax:

```
sagcc list license-tools keys [nodeAlias=nodeAlias] [productCode=productCode]
[productId=productId] [os=operatingSystemCode1,operatingSystemCode2...operatingSystemCodeN]
[release=releaseVersion] [platform=platformCode1,platformCode2...platformCodeN]
[installationType={Production|Development|Test}] [licenseKeyType={Standard|Custom}]
[aliasFilter=filter] [excludeExpired=true|false] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
[nodeAlias=nodeAlias]	Optional. The alias of the node for which to retrieve license keys. The command lists only the license keys that match the operating system of the node alias. The license keys with the same major version as the Platform Manager node are first of the list.
[productCode=productCode]	Optional. The product code of the product for which to retrieve license key information.

Argument or Option	Description
<code>[productId=<i>productId</i>]</code>	Optional. The ID of the product for which to retrieve license key information.
<code>[os=<i>operatingSystemCode1</i>, <i>operatingSystemCode2</i>... <i>operatingSystemCodeN</i>]</code>	Optional. The operating system or systems for which to retrieve license key information.
<code>[release=<i>releaseVersion</i>]</code>	Optional. The release version of the product for which to retrieve license keys.
<code>[platform=<i>platformCode1</i>, <i>platformCode2</i>...<i>platformCodeN</i>]</code>	Optional. The operating system platform or platforms for which to retrieve license key information.
<code>[installationType={Production Development Test}]</code>	Optional. The type of installation for which to retrieve license key information. Values are: <ul style="list-style-type: none"> ■ Production ■ Development ■ Test
<code>[licenseKeyType={Standard Custom}]</code>	Optional. The type of license key for which to retrieve information. Values are: <ul style="list-style-type: none"> ■ standard ■ custom
<code>[aliasFilter=<i>filter</i>]</code>	Optional. A search string that filters the license keys by aliases matching the string.
<code>[excludeExpired=true false]</code>	Optional. Specifies whether to include expired license keys in the search results. When set to <code>true</code> , the expired license keys are not included in the results. Default: <code>false</code>
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- The license key manager supports two types of license keys:
 - **Standard.** License keys for Software AG products that are produced and managed using standard Software AG licensing tools and processes. The standard keys are in .xml format.
 - **Custom.** License keys for Software AG products that have not yet adopted the standard Software AG licensing tools and processes, for example Terracotta. The custom keys are in any format that a product currently supports, for example a text file.
- You can use filters with the `sagcc list license-tools keys` command to retrieve information only about standard license keys.
- The command sorts the license keys in the search results in the following order:
 1. By type. The custom license keys are filtered by their alias and are always last on the list.
 2. By expiration date. Unlimited licenses are first on the list and expired licenses are last.
 3. If `nodeAlias` is included, the command determines which license keys fit best the specified node alias on the following criteria: core license metric, platform, and version. When the license keys are even on all criteria, the command sorts them by product name and key alias.

If `nodeAlias` is not included, the command sorts by product name, version, core license metric, and key alias.

Examples When Executing on Command Central

- To list all license keys for Integration Server:

```
sagcc list license-tools keys productCode=PIE
```

OR

```
sagcc list license-tools keys productId=integrationServer
```
- To list all license keys for Windows 7 and 8 operating systems:

```
sagcc list license-tools keys os=win7,win8
```
- To list all Windows license keys for Integration Server:

```
sagcc list license-tools keys productCode=PIE platform= W32,W64,WNT
```
- To list all license keys for Universal Messaging for the installation type “Production” on a MacOS operating system:

```
sagcc list license-tools keys productCode=NUM os=MacOS  
installationType=Production
```

sagcc update license-tools keys

Replaces the license key file for an existing license key alias with a new license key file.

Syntax

- Command Central syntax:

```
sagcc update license-tools keys licenseKeyAlias [--input|-i] filename [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>licenseKeyAlias</i>	Required. The existing license key alias to which to assign a new license key file.
{--input -i} <i>filename</i>	Required. Specifies the name of the input file. For more information, see "input" on page 177 .
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

The license key manager supports two types of license keys:

- **Standard.** License keys for Software AG products that are produced and managed using standard Software AG licensing tools and processes. The standard keys are in .xml format.
- **Custom.** License keys for Software AG products that have not yet adopted the standard Software AG licensing tools and processes, for example Terracotta. The custom keys are in any format that a product currently supports, for example a text file.

Examples When Executing on Command Central

To add the license key file "PIE96WinDesktop.xml" with the alias "PIEWinDesktop" to the license key manager:

```
sagcc add license-tools keys PIEWinDesktop
-i c:\Licenses\PIE96WinDesktop.xml
```

and assign a new license key, "PIE98WinDesktop.xml", to the alias "PIEWinDesktop":

```
sagcc update license-tools keys PIEWinDesktop
-i c:\Licenses\PIE98WinDesktop.xml
```

sagcc delete license-tools keys

Deletes all available product license keys or a license key with the specified license key alias.

Syntax

- Command Central syntax:

```
sagcc delete license-tools keys [licenseKeyAlias] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>licenseKeyAlias</i>	Optional. The alias of the license key to delete.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Examples When Executing on Command Central

To delete a license key with the alias "Terracotta":

```
sagcc delete license-tools keys Terracotta
```

sagcc update configuration license

Updates the license key file assigned to the specified license key alias for a particular run-time component.

Syntax

- Command Central syntax:

```
sagcc update configuration license nodeAlias runtimeComponentId
configurationInstanceId licenseKeyAlias [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. Specifies the alias name of the installation in which the run-time component is installed.
<i>runtimeComponentId</i>	Required. Specifies the run-time component for which you want to update the license key file.
<i>configurationInstanceId</i>	Required. Specifies the configuration instance that belongs to the run-time component. The only configuration instances that can be updated are the configuration instances that belong to a run-time component with the COMMON-LICENSE configuration type.
<i>licenseKeyAlias</i>	The alias of the license key file to update.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

Before you can execute the `sagcc update configuration license` command, you must add the license key file that you want to update to the license key manager using the `sagcc add license-tools keys` command. For more information about using the `sagcc add license-tools keys` command, see ["sagcc add license-tools keys" on page 320](#).

Examples When Executing on Command Central

To add the license key file "PIE96WinDesktop.xml" with the alias "PIEWinDesktop" to the license key manager:

```
sagcc add license-tools keys PIEWinDesktop -i c:\Licenses\PIE96WinDesktop.xml
```

and update the license key file with the alias "PIEWinDesktop" for the run-time component "IntegrationServer-default" installed on a node with the alias "sag01":

```
sagcc update configuration license sag01
IntegrationServer-default COMMON-LICENSE-IS-Core PIEWinDesktop
```


16 License Reports Commands

■ sagcc create license-tools reports snapshot	331
■ sagcc delete license-tools reports snapshot	331
■ sagcc delete license-tools reports snapshot reportid	332
■ sagcc list license-tools reports snapshot	332
■ sagcc get license-tools reports snapshot reportid	333
■ sagcc get license-tools reports snapshot output PDF	334
■ sagcc get license-tools reports snapshot output XML	335
■ sagcc create license-tools reports installation	336
■ sagcc delete license-tools reports installation reportid	336
■ sagcc delete license-tools reports installation	337
■ sagcc get license-tools reports installation output PDF	337
■ sagcc get license-tools reports installation output XML	338
■ sagcc get license-tools reports installation disclaimer	339
■ sagcc list license-tools reports installation	340
■ sagcc list license-tools reports	340
■ sagcc list license-tools metering products	341
■ sagcc add license-tools manifests	342
■ sagcc delete license-tools manifests	342
■ sagcc get license-tools manifests output xml	343
■ sagcc list license-tools manifests	344
■ sagcc list license-tools manifests content	344
■ sagcc list license-tools reports aggregated	346
■ sagcc get license-tools reports aggregated reportid	347
■ sagcc add license-tools reports aggregated	348
■ sagcc delete license-tool reports aggregated reportid	349
■ sagcc delete license-tools reports aggregated	350

■ sagcc exec license-tools metering	350
■ sagcc get license-tools metering state	351
■ sagcc get license-tools metering contracts licensekey	352
■ sagcc get license-tools reports aggregated usage reportid	353
■ sagcc list license-tools manifests contracts	354

sagcc create license-tools reports snapshot

Creates a license inventory report by license key based on the currently registered nodes in a Command Central instance.

Syntax

- Command Central syntax:

```
sagcc 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 "Common Options" on page 167.

Usage Notes

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

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

sagcc delete license-tools reports snapshot

Deletes all generated license reports by license key from Command Central.

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports snapshot [options]
options :
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

sagcc delete license-tools reports snapshot reportid

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

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports snapshot reportid [options]
  options :
  [--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code><i>reportid</i></code>	Required. The ID of the license report to delete.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To delete a license report with report ID `c4eff6`:

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

sagcc list license-tools reports snapshot

Lists all license reports by license key available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc list 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 "Common Options" on page 167.

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 inventory status of the landscape.
- When an existing report fails checksum verification, the report is not included in the output list.

sagcc get license-tools reports snapshot reportid

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

Syntax

- Command Central syntax:

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

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the report for which to obtain information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Examples When Executing on Command Central

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

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

sagcc get license-tools reports snapshot output PDF

Generates a PDF file for an existing license report by license key.

Syntax

- Command Central syntax:

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

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	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- <i>filename.pdf</i>	Required. Specifies a name for the output PDF file. For more information, see "output" on page 182.

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:

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

sagcc get license-tools reports snapshot output XML

Generates an existing license report by license key in XML format.

Syntax

- Command Central syntax:

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

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the license report for which to generate an XML file.
<code>--output-format xml</code>	Required. Specifies that the output is generated in XML format.
<code>--output-filename.xml</code>	Required. Specifies a name for the output XML file. For more information, see "output" on page 182.

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:

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

sagcc create license-tools reports installation

Creates a license report of the current landscape based on a license manifest file.

Syntax

- Command Central syntax:

```
sagcc create license-tools reports installation [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 "Common Options" on page 167.

Usage Notes

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

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

sagcc delete license-tools reports installation reportid

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

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports installation reportid [options]
options :
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the license report to delete.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To delete a report by license manifest with report ID `c4eff7`:

```
sagcc delete license-tools reports installation c4eff7
```

sagcc delete license-tools reports installation

Deletes all generated reports by license manifest from the Command Central server.

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports installation [options]  
options:  
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

sagcc get license-tools reports installation output PDF

Generates a PDF file for an existing license report by license manifest.

Syntax

- Command Central syntax:

```
sagcc get license-tools reports installation reportid --output-format pdf
--output filename.pdf
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	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 <i>filename.pdf</i>	Required. Specifies a name for the output PDF file. For more information, see "output" on page 182.

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 `c4eff7` and save the generated report file in the current directory:

```
sagcc get license-tools reports installation c4eff7 --output-format pdf
--output report.pdf
```

sagcc get license-tools reports installation output XML

Generates an existing license report by license manifest in XML format.

Syntax

- Command Central syntax:

```
sagcc get license-tools reports installation reportid --output-format xml
--output filename.xml
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the license report for which to generate an XML file.
<code>--output-format xml</code>	Required. Specifies that the output is generated in XML format.
<code>--output-filename.xml</code>	Required. Specifies a name for the output XML file. For more information, see "output" on page 182.

Example When Executing on Command Central

To generate an XML file, named `report.xml`, for a license report with ID `c4eff7` and save the generated report file in the current directory:

```
sagcc get license-tools reports installation c4eff7 --output-format xml
--output report.xml
```

sagcc get license-tools reports installation disclaimer

Displays the disclaimer text that is valid for the reports by license manifest.

Syntax

- Command Central syntax:

```
sagcc get license-tools reports installation disclaimer [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

sagcc list license-tools reports installation

Lists all reports by license manifest available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc list license-tools reports installation [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 "Common Options" on page 167.

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 inventory status of the landscape.

sagcc list license-tools reports

Lists all license reports, by license key, by license manifest, and aggregated reports, available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc list license-tools reports [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

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 inventory status of the landscape.

sagcc list license-tools metering products

Lists the names and codes of all Software AG products for which license reports are available.

Syntax

- Command Central syntax:


```
sagcc list license-tools metering products [options]
```
- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

sagcc add license-tools manifests

Adds a license manifest file with the specified manifest alias to the Command Central server.

Syntax

- Command Central syntax:

```
sagcc add license-tools manifests manifestAlias [--input|-i] filename.xml
[options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>manifestAlias</i>	Required. The alias for the license manifest file to add.
{--input -i} <i>filename.xml</i>	Required. Specifies the name of the input file. For more information, see "input" on page 177.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To add license manifest file "Manifest1" with alias "Alias1":

```
sagcc add license-tools manifests alias1 --input manifest1.xml
```

sagcc delete license-tools manifests

Deletes all available license manifest files or a license manifest with the specified manifest alias.

Syntax

- Command Central syntax:

```
sagcc delete license-tools manifests [manifestAlias] [options] options :
[--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>manifestAlias</i>	Optional. The alias for the license manifest file to delete.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To delete a license manifest with alias "Alias1":

```
sagcc delete license-tools manifests alias1
```

sagcc get license-tools manifests output xml

Generates an existing license manifest file in XML format.

Syntax

- Command Central syntax:

```
sagcc get license-tools manifests manifestAlias --output-format xml
--output filename.xml[options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>manifestAlias</i>	Required. The alias of the license manifest for which to generate an XML file.
--output-format xml	Required. Specifies that the output is generated in XML format.
--output- <i>filename.xml</i>	Required. Specifies a name for the output XML file. For more information, see "output" on page 182.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a

Argument or Option	Description
	description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To generate an XML file, named manifest1.xml, for a license manifest with alias "Alias1":

```
sagcc get license-tools manifests alias1 --output-format xml
--output manifest1.xml
```

sagcc list license-tools manifests

Lists all license manifests available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc list license-tools manifests [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 "Common Options" on page 167.

sagcc list license-tools manifests content

Lists the details available in a specified license manifest. Information about the licensed units in the license manifest can include:

- Type: contract, bundle, or product
- Name
- Version
- Operating System
- Number of licensed units
- Type of installation

- Whether metering is enabled for a licensed unit
- Status: pending, active, or expired

Syntax

- Command Central syntax:

```
sagcc list license-tools manifests content manifestAlias [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>manifestAlias</i>	Required. The alias of the license manifest from which you want to retrieve details.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Usage Notes

In the Operating System column in the command output, the codes for operating systems match the codes for supported operating systems in your product licensing agreement. The following table maps the operating system codes from the product licensing agreement to the operating system codes used by Command Central:

Operating System	OS code in licensing agreement	OS code in Command Central
AIX	VO	AIX
CentOS Linux	YT	LNAMD64
HP-UX	V7	HP11
MAC OS	V8	OSX
OS/400 RISC	W9	AS400
Oracle Linux	YO	LNAMD64
Red Hat Linux	YC	LNAMD64

Operating System	OS code in licensing agreement	OS code in Command Central
Red Hat Linux (zSeries)	ZV	LNXS390X
SUSE Linux	YB	LNXAMD64
SUSE Linux (zSeries)	ZW	LNXS390X
Solaris Ultra SPARC	ZO	SOL
Ubuntu	YU	LNXAMD64
Windows	Y8	W64
Windows Desktop	YK	W64
Windows Servers	YL	W64

Example

To retrieve the details about the license units included in the license manifest with alias "MyLicenseManifest":

```
sagcc list license-tools manifests content MyLicenseManifest
```

sagcc list license-tools reports aggregated

Lists all aggregated reports available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc list license-tools reports aggregated [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

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 date the report was last created, and the inventory status of the landscape.

sagcc get license-tools reports aggregated reportid

Generates a file for an existing aggregated report in PDF, XML, or JSON format.

Syntax

- Command Central syntax:

```
sagcc get license-tools reports aggregated reportid
--output-format.{pdf|xml|json} --output filename.{pdf|xml|js}
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the aggregated report for which to generate a PDF, XML, or JSON file. The report ID is case-sensitive.
<code>--output-format.{pdf xml json}</code>	Required. Specifies the format of the generated output. For more information, see "output-format" on page 183.

Argument or Option	Description
<code>--output filename. {pdf xml js}</code>	Required. Specifies a name for the output file. For more information, see "output" on page 182.

Usage Notes

You can archive an aggregated report by downloading it in XML format. You can later add the report back to Command Central, using the following command:

```
sagcc add license-tools reports aggregated [--input|-i] filename.xml
```

Examples When Executing on Command Central

- To archive an aggregated report with ID `Jun2015` in the current directory:

```
sagcc get license-tools reports aggregated Jun2015  
--output-format xml --output Jun2015.xml
```

To then add the archived aggregated report in Command Central:

```
sagcc add license-tools reports aggregated -i Jun2015.xml
```

- To generate a PDF file for an aggregated report with ID `Jun2015` and save the generated report file in the current directory:

```
sagcc get license-tools reports aggregated Jun2015  
--output-format pdf --output Jun2015.pdf
```

- To generate a JSON file for an aggregated report with ID `Jun2015` and save the generated report file in the current directory:

```
sagcc get license-tools reports aggregated Jun2015  
--output-format json --output Jun2015.js
```

sagcc add license-tools reports aggregated

Adds an archived aggregated report in XML format to Command Central.

Syntax

- Command Central syntax:

```
sagcc add license-tools reports aggregated [--input|-i] filename.xml [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>{--input -i} filename.xml</code>	Required. Specifies the name of the input file. For more information, see "input" on page 177.

Argument or Option	Description
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Usage Notes

You use the `sagcc add license-tools reports aggregated` command to add an aggregated report that was archived using the following command:

```
sagcc get license-tools reports aggregated reportid --output-format xml
--output filename.xml
```

Example When Executing on Command Central

To archive an aggregated report with ID `Jun2015`:

```
sagcc get license-tools reports aggregated jun2015
--output-format xml --output Jun2015.xml
```

To then add the archived aggregated report in Command Central:

```
sagcc add license-tools reports aggregated -i Jun2015.xml
```

sagcc delete license-tool reports aggregated reportid

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

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports aggregated reportid [options]
  options:
  [--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the aggregated report to delete. The report ID is case-sensitive.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To delete an aggregated report with report ID Jun2015:

```
sagcc delete license-tools reports aggregated Jun2015
```

sagcc delete license-tools reports aggregated

Deletes all aggregated reports available on the Command Central server.

Syntax

- Command Central syntax:

```
sagcc delete license-tools reports aggregated [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 "Common Options" on page 167.

sagcc exec license-tools metering

Starts or stops continuous license metering of a Command Central landscape. Command Central meters the landscape every full hour and stores the results in a monthly aggregated report. Depending on the size of the landscape, the metering and hourly report creation may take between several seconds and several minutes.

Syntax

- Command Central syntax:

```
sagcc exec license-tools metering action [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>action</i>	<p>Required. Specifies the action you want to take against continuous license metering of a landscape. Specify one of the following actions:</p> <ul style="list-style-type: none"> ■ <code>start</code> - Starts continuous license metering and report aggregation for a landscape. ■ <code>stop</code> - Stops continuous license metering and report aggregation for a landscape.
<i>[options]</i>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

To check whether continuous license metering of a landscape is underway, run the following command:

```
sagcc get license-tools metering state
```

sagcc get license-tools metering state

Returns the license metering state for a landscape. Continuous license metering of a landscape is either in started or stopped state.

Syntax

- Command Central syntax:

```
sagcc get license-tools metering state [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>[options]</i>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

sagcc get license-tools metering contracts licensekey

Returns all automatic assignments of a run-time component to a license, or a list of assignments based on the following filters:

- Node alias
- Run-time component ID

Syntax

- Command Central syntax:

```
sagcc get license-tools metering contracts licensekey
[nodeAlias=nodeAlias] [runtimeComponentId=runtimeComponentId] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
[nodeAlias=nodeAlias]	Optional. Specifies the alias name of the installation in which the run-time component is installed.
[runtimeComponentId=runtimeComponentId]	Optional. Specifies the run-time component for which to retrieve the automatic assignment of a license.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

You use the command to retrieve the following information:

- The automatic assignment of a license for run-time components that have a license key attached.
- Whether the identified license of each assignment can be found in an active license manifest file.
 - If the license can be found in an active license manifest file, the command returns a contract item identifier that contains the location ID of the manifest.
 - If the license cannot be found in an active license manifest file, the command returns a contract item identifier that has a null location ID.

Example When Executing on Command Central

To retrieve the automatic license assignment for the run-time component “IntegrationServer_default” installed in the installation with alias name “sag01”:

```
sagcc get license-tools metering contracts licensekey nodeAlias=sag01
runtimeComponentId=IntegrationServer_default
```

sagcc get license-tools reports aggregated usage reportid

Generates a file in PDF, XML, or JSON format that contains metered data based on the total product usage per month and the base and peak licenses. Command Central extracts the data for the aggregated usage report from an existing aggregated license report.

An aggregated license report contains the metered data for all installed products and all licenses in a license manifest file.

Syntax

- Command Central syntax:

```
sagcc get license-tools reports aggregated usage reportid
--output-format.{pdf|xml|json} --output filename.{pdf|xml|js}
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>reportid</i>	Required. The ID of the license report for which to generate a PDF, XML, or JSON file. The report ID is case-sensitive.
<code>--output-format.{pdf xml json}</code>	Required. Specifies the format of the generated output. For more information, see "output-format" on page 183.
<code>--output filename.{pdf xml js}</code>	Required. Specifies a name for the output file. For more information, see "output" on page 182.

Examples When Executing on Command Central

- To generate a PDF file for a usage report with ID Jun2015 and save the generated report file in the current directory:

```
sagcc get license-tools reports aggregated usage Jun2015
--output-format pdf --output Jun2015.pdf
```

- To generate an XML file for a usage report with ID `Jun2015` and save the generated report file in the current directory:

```
sagcc get license-tools reports aggregated usage Jun2015
--output-format xml --output Jun2015.xml
```

- To generate a JSON file for a usage report with ID `Jun2015` and save the generated report file in the current directory:

```
sagcc get license-tools reports aggregated usage Jun2015
--output-format json --output Jun2015.js
```

sagcc list license-tools manifests contracts

Lists all available licenses for the product with the specified product ID.

Syntax

- Command Central syntax:

```
sagcc list license-tools manifests contracts nodeAlias productId
[options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. Specifies the alias name of the installation in which the product with the specified product ID is installed.
<i>productId</i>	Required. Specifies the ID of the product for which you want to list the available licenses. You can determine the ID of a product using the <code>sagcc list inventory products</code> command.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.

Example When Executing on Command Central

To list the available licenses for a product with ID "integrationServer" installed in the installation with alias name "sag01":

```
sagcc list license-tools manifests contracts sag01 integrationServer
```

17 Lifecycle Commands

■ sagcc exec lifecycle	356
■ Specifying Search Criteria for Lifecycle Commands	359

sagcc 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:

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

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

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

■ Platform Manager syntax:

```
sagcc 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
<i>action</i>	<p>Required. Specifies the action you want to take against the run-time component. Supply one of the following actions:</p> <ul style="list-style-type: none"> ■ <i>start</i> - starts the run-time component ■ <i>startindebugmode</i> - starts the run-time component in debug mode ■ <i>startinsafemode</i> - starts the run-time component in safe mode ■ <i>stop</i> - stops the run-time component ■ <i>restart</i> - stops, then restarts the run-time component

Argument or Option	Description
	<ul style="list-style-type: none"> ■ <code>pause</code> - pauses the run-time component ■ <code>resume</code> - resumes previously paused run-time component <p>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.</p>
<code>node_alias</code>	<p>Command Central only.</p> <p>Required when you do not specify search criteria. Specifies the alias name of an installation. You can determine installation alias names using the sagcc list landscape nodes command.</p>
<code>componentid</code>	<p>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 sagcc list inventory components command.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p>Note: Command Central only requires the <code>componentid</code> when you do not specify search criteria.</p> <p>When executing the command against a Platform Manager, specify the <code>componentid</code> before the action in the command syntax.</p> </div>
[<code>criteria</code>]	<p>Command Central only.</p> <p>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 359.</p>
[<code>options</code>]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

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

- You can execute the `sagcc get monitoring runtimestatus` command after executing the `sagcc 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 `sagcc get monitoring runtimestatus` command to specify the results for which to check and how often to check for the results. For more information, see ["sagcc get monitoring" on page 362](#).
- You cannot stop the OSGI-SPM component using the `sagcc exex lifecycle` command.

Examples When Executing on Command Central

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

```
sagcc 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" on page 189](#), ["username" on page 192](#), and ["password" on page 186](#).

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

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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":

```
sagcc 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":

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

Specifying Search Criteria for Lifecycle Commands

When executing the `sagcc 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] attribute1=value1 attribute2=value2 ...
```

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

Attribute Names You Can Use in the Search Criteria

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

You can combine any of the attribute names in this list with new search attributes added using the `sagcc create inventory components attributes` command. For more information about syntax and usage, see [sagcc create inventory components attributes](#).

Logical Operators Used When Specifying Multiple Search Attributes

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 attributes. 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
```

18 Monitoring Commands

■ sagcc get monitoring	362
■ sagcc get monitoring state	364
■ sagcc list monitoring alerts	367

sagcc get monitoring

DEPRECATED. Replaced by the [sagcc get monitoring state](#) command. 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:

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

- Platform Manager syntax:

```
sagcc 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
<code>{runtimestatus runtimestate state}</code>	Required. Specifies whether you want to retrieve run-time statuses, run-time states, or state.
<code>node_alias</code>	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 sagcc list landscape nodes .
<code>componentid</code>	Required for <code>runtimestatus</code> and <code>runtimestate</code> . Not applicable for <code>state</code> . 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 sagcc list inventory components .
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

Usage Notes

- The following are general descriptions of run-time statuses:
 - FAILED when a run-time component failed, for example, ended unexpectedly.
 - NOT_READY when a run-time component is started, but not ready to accept client requests.
 - 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:

```
sagcc 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" on page 189](#) and ["username" on page 192](#). 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 `sagcc 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:

```
sagcc exec lifecycle stop sag01 OSGI-IS --password secret

sagcc 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" on page 189](#) and ["username" on page 192](#). 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:

```
sagcc 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" on page 192](#). The command specifies “secret” for the user’s password.

sagcc get monitoring state

Retrieves the run-time status and run-time state of a run-time component.

- Run-time status indicates whether a run-time component is running or not.
- Run-time state indicates the health of a run-time component by providing (key performance indicators (KPIs) for the component.

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

Syntax

■ Command Central syntax:

```
sagcc get monitoring state [nodeAlias=alias ]
[runtimeComponentId=componentid ] [includeChildren=true]
[refresh=true] [options]
```

■ Platform Manager syntax:

```
sagcc get monitoring state [runtimeComponentId=componentid ] [refresh=true]
[options]
```

Arguments and Options

Argument or Option	Description
[nodeAlias= <i>alias</i>]	<p>Command Central only.</p> <p>Optional. Specifies the alias name of the installation on which the run-time component is installed.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>
[runtimeComponentId= <i>componentid</i>]	<p>Optional. Specifies the ID of the run-time component for which you want to retrieve information.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p>
[includeChildren=true]	<p>Command Central only.</p> <p>Optional. Indicates whether to retrieve the status and state for a run-time component and its dependent components.</p> <p>When you include this argument, Command Central retrieves monitoring data for the run-time component and all of its dependent components.</p> <p>When you do not include this argument, Command Central retrieves monitoring data <i>only</i> for the run-time component.</p>
[refresh=true]	<p>Optional.</p> <ul style="list-style-type: none"> ■ On Command Central: <p>When you include this argument, Command Central retrieves monitoring data from the</p>

Argument or Option	Description
	<p>Platform Manager node. When you do not include this argument, Command Central retrieves monitoring data from its cache.</p> <ul style="list-style-type: none"> ■ On Platform Manager: <p>When you include this argument, Platform Manager requests the current monitoring state directly from the product.</p>
[options]	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- The following are general descriptions of run-time statuses:
 - FAILED when a run-time component failed, for example, ended unexpectedly.
 - NOT_READY when a run-time component is started, but not ready to accept client requests.
 - 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:

```
sagcc get monitoring state nodeAlias=sag01
runtimeComponentId=OSGI-SPM refresh=true --format xml --password secret
```

Because the command includes the `refresh=true` parameter, the monitoring data will be retrieved from the Platform Manager node.

- To get the run-time status and run-time state for a component with ID "OSGI-IS" running in the installation "sag01", and all its dependent components:

```
sagcc get monitoring state nodeAlias=sag01 runtimeComponentId=OSGI-IS
includeChildren=true
```

Because the command does not include the `refresh=true` parameter, the monitoring data will be retrieved from the Command Central cache.

Example When Executing on Platform Manager

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

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

sagcc list monitoring alerts

Lists the alerts for a specified run-time component.

Syntax

- Command Central syntax:

```
sagcc list monitoring alerts [nodeAlias=alias]
[runtimeComponentId=componentid] [includeChildren=true] [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
[nodeAlias= <i>alias</i>]	Optional. 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 sagcc list landscape nodes .
[runtimeComponentId= <i>componentid</i>]	Optional. The ID of the run-time component for which you want to retrieve information. You can determine the IDs for run-time components using sagcc list inventory components .

Argument or Option	Description
[includeChildren=true]	<p>Optional. Indicates whether to retrieve the status and state for a run-time component and its dependent components.</p> <p>When you include this argument, Command Central retrieves monitoring data for the run-time component and all of its dependent components.</p> <p>When you do not include this argument, Command Central retrieves monitoring data <i>only</i> for the run-time component.</p>
[options]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167.</p>

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:

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

19 Provisioning Assets Commands

■ sagcc exec provisioning assets install	370
■ sagcc exec provisioning assets uninstall	372
■ sagcc list provisioning assets supportedcomponents	373

sagcc exec provisioning assets install

Installs assets from an asset repository.

Important: The assets provisioning commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

```
sagcc exec provisioning assets nodeAlias repoName install
runtimeComponentId=id artifacts=[AssetId1,
AssetId2 |ALL] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the target node on which to install assets.
<i>repoName</i>	Required. The name of the asset repository from which to install assets.
runtimeComponentId= <i>id</i>	Required. The ID of the run-time component for which you want to install assets.
artifacts=[<i>AssetId1</i> , <i>AssetId2</i> ALL]	A comma-separated list of the IDs of the assets you want to install. When you set artifacts=ALL or omit the artifacts argument altogether, the command installs all assets that belong to the run-time component you specify in the runtimeComponent argument.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface.

Argument or Option	Description
	For a description of the options, see "Common Options" on page 167 .

Usage Notes

- Command Central supports installing *only* composite assets.
- You can install assets for some run-time components in installations with release versions lower than 10.1. To check which products in an installation support installing assets, run:


```
sagcc list provisioning assets nodeAlias supportedcomponents
```

 and replace `nodeAlias` with the alias name of the installation. For details about the command, see ["sagcc list provisioning assets supportedcomponents" on page 373](#).
- Command Central identifies each runtime, for which to install assets, by host, port, and runtime type. The host is resolved from the values of the `nodeAlias` and `runtimeComponentId` arguments. The type of the runtime is obtained by mapping the `runtimeComponentID` to a run-time type as described in ["sagcc list provisioning assets supportedcomponents" on page 373](#).
- Command Central identifies the endpoint port for a runtime using the following criteria, ordered in descending priority:
 1. Is "primary" property set to true
 2. Is "protocol" property set to HTTP
 3. Is "protocol" property set to HTTPS
 4. Port number: the lower port number is selected first
 For example, if a run-time component has the following set of ports: (`isPrimary, http, 5555`) and (`!isPrimary, http, 8072`), Command Central selects port 5555, because "isPrimary" is the criteria with the highest priority.
- When installing assets, you can check the progress of the install operation and troubleshoot issues using the CLI `jobsmanager jobs` commands or in the Command Central web user interface, go to `Stacks > Jobs`.

Examples

To install all composite assets for the run-time component with ID "MwsProgramFiles-default", which are available in the asset repository with name "repo1", on the target installation with alias name "sag01":

```
sagcc exec provisioning assets sag01 repo1 install
runtimeComponentId=MwsProgramFiles-default
```

sagcc exec provisioning assets uninstall

Uninstall assets for a specified run-time component.

Important: The assets provisioning commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

```
sagcc exec provisioning assets nodeAlias uninstall
runtimeComponentId=id artifacts=[AssetId1,
AssetId2] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the target node from which to uninstall assets.
runtimeComponentId= <i>id</i>	Required. The ID of the run-time component for which you want to uninstall assets.
artifacts=[<i>AssetId1</i> , <i>AssetId2</i>]	Required. A comma-separated list of the IDs of the assets you want to uninstall. Command Central uninstalls only the assets listed in the <code>artifacts</code> argument and does not support uninstalling all assets that belong to a run-time component.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

Uninstalling an asset removes the asset from the target installation and does not perform a roll-back to the previous version of the asset.

Examples When Executing on Command Central

To uninstall the asset with ID "Fabonachi" for the run-time component with ID "MwsProgramFiles-default" from the target installation with alias name "sag01":

```
sagcc exec provisioning assets sag01 uninstall
runtimeComponentId=MwsProgramFiles-default artifacts=Fabonachi
```

sagcc list provisioning assets supportedcomponents

Retrieves a list of run-time components for which you can install assets and their corresponding run-time types.

Important: The assets provisioning commands are a preview feature that is subject to change in the future. This preview has limited functions and is not intended for use in a production environment. If you want to provide feedback for this preview feature, go to the Command Central area in the Software AG TechCommunity.

Syntax

- Command Central syntax:

```
sagcc list provisioning assets nodeAlias supportedcomponents
[options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the installation for which to retrieve information. You can view a list of installations and their aliases using the <code>sagcc list landscape nodes</code> command.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- The command output returns the mapping of the run-time components to run-time types, for example:

Runtime Type	Runtime Component Id
EDA	OSGI-SPM-EDA-DEPLOYER
BPM	OSGI-IS_default-WmPRT
RULES	OSGI-IS_default-WmBusinessRules
DES	OSGI-SPM-DES-DEPLOYER
MWS	MwsProgramFiles-default
BPM	OSGI-IS_default-WmDBO
IS	integrationServer-default

- The command output lists only the run-time components of products that support installing assets with Command Central and are installed in the specified installation.

Examples

- To list the run-time components that support installing assets in the installation of release 9.12 and alias name "remote_9.12":

```
sagcc list provisioning assets remote_9.12 supportedcomponents
```

The command returns the following output, which indicates that in this installation only the "integrationServer-default" run-time component supports installing assets:

Runtime Type	Runtime Component Id
IS	integrationServer-default

- To retrieve a list of the run-time components in the installation with alias name "local" for which you can install assets:

```
sagcc list provisioning assets local supportedcomponents
```

20 Provisioning Bootstrap Installers Commands

■ sagcc create provisioning bootstrap installers	376
■ sagcc list provisioning bootstrap installers	378
■ sagcc exec provisioning bootstrap nodes	380

sagcc create provisioning bootstrap installers

DEPRECATED. Creates a bootstrap installer image from a local or remote master product repository, configured with valid user credentials, and registers the bootstrap installer image as an image product repository.

Use the Command Central bootstrap installer to bootstrap Platform Manager installations for releases 9.12 and higher. You can download the Command Central bootstrap installer for the release that you want from the Empower Product Support website. Save the downloaded Command Central bootstrap installer file into the `$CC_HOME/profiles/CCE/data/installers` directory, so that you can use the installer in the `nodes` section of the composite templates, which is the recommended way to bootstrap nodes.

Syntax

- Command Central syntax:

- With an input file:

```
sagcc create provisioning bootstrap installers
  {--input | -i} filename{.xml | .json} [options]
```

- With arguments:

```
sagcc create provisioning bootstrap installers [version=version]
  [repoName= name] [platform=platform] [product=product]
  [distribution=type] [options]
```

- Not supported on Platform Manager.

Options

Option	Description
<code>{--input -i}</code> <code>filename{.xml .json}</code>	<p>Optional. Input file in XML or JSON format that contains the bootstrap installer image details. For more information, see "input" on page 177.</p> <p>You can either use an input data file or specify arguments for the command. For an example of a bootstrap details XML file, see the usage notes for the sagcc exec provisioning bootstrap nodes command.</p>
<code>[version=version]</code>	<p>Version of the product repository for which to create the bootstrap installer image.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> ■ <i>major</i> version (for example, 9)

Option	Description
	<ul style="list-style-type: none"> ■ <i>major.minor</i> version (for example, 9.8) ■ <i>latest</i> <p>If Command Central cannot match the specified version to a registered product repository, Command Central returns an error.</p> <p>If you do not specify this option, the command uses the latest version available in the product repository.</p>
[repoName= <i>name</i>]	<p>Name of the product repository from which to create the bootstrap installer image.</p> <p>If you do not include this parameter, the command uses:</p> <ul style="list-style-type: none"> ■ The product repository that matches the version specified in the <i>version=version</i> argument ■ If the <i>version=version</i> argument is not specified, the registered repository for the latest version. ■ If only one product repository is registered in Command Central, the command uses that repository. <p>If no repositories are registered in Command Central, the command returns an error.</p>
[platform= <i>platform</i>]	<p>ID of the target operating system for which to create the bootstrap installer image.</p> <p>If you do not specify this option, the command uses the operating system on the localhost machine.</p>
[product= <i>product</i>]	<p>ID of products to include in the bootstrap installer image. Valid values are SPM, CCE, and CLI.</p>
[distribution= <i>type</i>]	<p>Software distribution to include in the bootstrap installer image. Valid values are:</p> <ul style="list-style-type: none"> ■ MIN. Platform Manager only. Supported only for a local installation on the Command Central host. This distribution does not install the Java Runtime Environment (JRE) supported by

Option	Description
	<p>webMethods products and uses the JRE of the target machine.</p> <ul style="list-style-type: none"> ■ DEF. Platform Manager, Update Manager, and the Command Central command line interface. ■ ALL. Platform Manager, Update Manager, the Command Central command line interface, and all Platform Manager product plug-ins. Supported only for releases 9.8 and 9.9 <p>If you do not specify this option, the command includes the complete distribution.</p>
[options]	Any option supported by the command line interface (see " Common Options " on page 167).

Usage Notes

If you want to create the bootstrap installer image from a mirror repository, you must specify only mirror repositories created on the local Platform Manager installation.

Examples When Executing on Command Central

- To create a bootstrap installer image with complete distribution from the latest product version for the operating system on the localhost machine:

```
sagcc create provisioning bootstrap installers
```

- To create a bootstrap installer image with complete distribution from a repository with version "9.8" for operating system with ID "LNXAMD64":

```
sagcc create provisioning bootstrap installers
version=9.8 platform=LNXAMD64 distribution=ALL
```

- To create a bootstrap installer image, based on the "bootstrapInfo.xml" file that contains the bootstrap data required to create the image:

```
sagcc create provisioning bootstrap installers --input bootstrapInfo.xml
```

sagcc list provisioning bootstrap installers

Retrieves a list of available bootstrap installers.

Software AG recommends using the Command Central installers that you can download from the Empower Product Support website, instead of the installers created with the `sagcc create provisioning bootstrap installers` command.

After you download the Command Central bootstrap installer for the release that you want from the Empower Product Support website, save the downloaded Command Central bootstrap installer file into the `$CC_HOME/profiles/CCE/data/installers`

directory. The command lists all installers in this directory and you can use them to bootstrap local and remote nodes.

Syntax

- Command Central syntax:

- To download a Command Central bootstrap installer:

```
sagcc get provisioning bootstrap installers installerName {.sh | .zip}
-o installerName {.sh | .zip}
```

- To list the available Command Central bootstrap installers:

```
sagcc list provisioning bootstrap installers [version=version]
[platform=platformCode] [options]
```

- Not supported on Platform Manager.

Options

Option	Description
<i>installerName</i> { <i>.sh</i> <i>.zip</i> }	The name of the installer file to find (for example, <code>cc-def-9.12-lnxadm64.sh</code>).
<i>version=version</i>	The version of the bootstrap installer to find.
[<i>platform=platformCode</i>]	Code of the target operating system of the bootstrap installer to find.
[<i>product=SPM CCE</i>]	DEPRECATED argument, not required for the Command Central bootstrap installers. The ID of Command Central (CCE) or Platform Manager (SPM).
[<i>options</i>]	Any options supported by the command line interface (see " Common Options " on page 167).

Examples When Executing on Command Central

- To download the bootstrap installer with name "cc-def-9.12-lnxadm64.sh":

```
sagcc get provisioning bootstrap installers cc-def-9.12-lnxamd64.sh
-o cc-def-9.12-lnxamd64.sh
```

- To list all available bootstrap installers:

```
sagcc list provisioning bootstrap installers
```

- To list bootstrap installers for repository version "9.12" and operating system "OSX":

```
sagcc list provisioning bootstrap installers version=9.12
platform=OSX
```

sagcc exec provisioning bootstrap nodes

DEPRECATED. Installs a Platform Manager node on a local or remote machine and registers the new Platform Manager installation under a specified node alias.

Use the Command Central bootstrap installer to bootstrap Platform Manager installations for releases 9.12 and higher. You can download the Command Central bootstrap installer for the release that you want from the Empower Product Support website. Save the downloaded Command Central bootstrap installer file into the `$CC_HOME/profiles/CCE/data/installers` directory, so that you can use the installer in the `nodes` section of the composite templates to bootstrap remote Platform Manager installations.

Syntax

■ Command Central syntax:

■ With an input file:

```
sagcc exec provisioning bootstrap nodes nodeAlias
{--input | -i} filename{.xml | .json}
```

■ With arguments:

```
sagcc exec provisioning bootstrap nodes nodeAlias [hostname=hostname]
[repoName=name] [platform=platform] [installDir=directory]
[httpPort=port] [httpsPort=port] [options]
```

■ Not supported on Platform Manager.

Options

Option	Description
<code>{--input -i}</code> <code><i>filename</i>{.xml .json}</code>	Optional. Input file in XML or JSON format that contains bootstrap installation details. For more information, see "input" on page 177 . You can either use an input data file or specify arguments for the command. For an example of a bootstrap details XML file, see the Usage Notes section.
<code><i>nodeAlias</i></code>	Required. Alias name of the bootstrapped Platform Manager installation that you want to add to the target machine.
<code>[hostname=<i>hostname</i>]</code>	Optional. Host name of the machine on which to install the Platform Manager node. The default is <code>localhost</code> .

Option	Description
<code>[repoName=<i>name</i>]</code>	<p>Optional. Name of the product repository, registered in Command Central from which the bootstrap installer image is created.</p> <p>When you do not include this parameter, the command uses the product repository with the latest version.</p> <p>If no repositories are registered in Command Central, the command returns an error.</p>
<code>[platform=<i>platform</i>]</code>	<p>Optional. ID of the target operating system of the bootstrap installer.</p> <p>When you do not include this parameter, the command uses the operating system on the localhost machine.</p>
<code>[installDir=<i>directory</i>]</code>	<p>Optional. Directory in which to install the bootstrap Platform Manager installation. The value for the <code>installDir</code> argument is platform independent and Command Central resolves the format of the installation directory path based on the target operating system. For example, if you specify <code>installDir=/opt/softwareag</code>, Command Central resolves the format of the installation directory path to <code>C:\opt\softwareag</code> on Windows.</p>
<code>[httpPort=<i>port</i>]</code>	<p>Optional. HTTP port for the bootstrapped Platform Manager installation. The default is 8092.</p>
<code>[httpsPort=<i>port</i>]</code>	<p>Optional. HTTPS port for the bootstrapped Platform Manager installation. The default is 8093.</p>
<code>[options]</code>	<p>Optional. Any common option supported by the Command Line Interface (see "Common Options" on page 167)</p>

Usage Notes

- You can use this command to install Platform Manager only if the target UNIX or Windows operating systems have Open Secure Shell (OpenSSH) installed and are

configured for remote access. For information about how to install and configure OpenSSH on Windows operating systems, see the OpenSSH documentation.

- After you execute the `sagcc exec provisioning bootstrap nodes` command, use the `sagcc get landscape nodes` command to verify the node status. For example:

```
sagcc get landscape nodes sag01 --expected-values ONLINE
--wait 60 --check-every 5
```

where “sag01” is the alias name of the bootstrapped Platform Manager installation.

When the execution of the `sagcc get landscape nodes` command is successful, you can apply templates onto the new Platform Manager installation. For more information about the `--expected-values`, `--wait`, and `--check-every` options see ["expected-values" on page 173](#), ["wait" on page 193](#), and ["check-every" on page 168](#).

- Following is an example of a bootstrap details XML file required for bootstrapping a Platform Manager installation:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <bootstrapInfo>
    <repoName>webMethods-9.12_US</repoName>
    <version>latest.integration</version>
    <platform>OSX</platform>
    <distribution>DEF</distribution>
    <useImage>true</useImage>
    <hostname>localhost</hostname>
    <protocol>SSH</protocol>
    <port>22</port>
    <credentials>
      <authenticationType>BASIC</authenticationType>
      <authenticationMethod>INTERACTIVE</authenticationMethod>
      <userName>Administrator</userName>
      <password>manage</password>
    </credentials>
    <installDir>/some/dir</installDir>
    <installJava>>false</installJava>
    <httpPort>1234</httpPort>
    <httpsPort>1235</httpsPort>
  </bootstrapInfo>
```

Valid values for the `<authenticationMethod>` parameter are `PASSWORD` (default), `INTERACTIVE`, and `CERTIFICATE`.

Examples When Executing on Command Central

- To install a Platform Manager from the latest version of a bootstrap image generated from a node with alias name “sag01”, HTTP port “8092”, and HTTPS port “8093”, in the “c:\install\test” directory on a local node:

```
sagcc exec provisioning bootstrap nodes sag01
installDir=c:\install\test httpPort=8092 httpsPort=8093
```

- To install a Platform Manager from a bootstrap image with alias name “sag01”, default authentication credentials, and default installation details (such as ports, installation directory, and distribution type) on a remote node with host name “rubicon” and operating system “LNXADM64”:

```
sagcc exec provisioning bootstrap nodes sag01 hostname=rubicon  
platform=LNxADM64
```

- Bootstrap installation details are in the bootstrapInfo.xml file. To install a Platform Manager from a bootstrap image with alias name “sag01”, using the details in the bootstrapInfo.xml file:

```
sagcc exec provisioning bootstrap nodes sag01 --input bootstrapInfo.xml
```


21 Provisioning Fixes Commands

- `sagcc exec provisioning fixes install` 386
- `sagcc exec provisioning fixes uninstall` 387

sagcc exec provisioning fixes install

Installs fixes from a fix repository. The Platform Manager on which you install the fixes must have access to a fix repository.

Before installing fixes using this command, see ["Before Installing Fixes and Support Patches" on page 43](#).

Syntax

- Command Central syntax:

```
sagcc exec provisioning fixes node_alias repo_name install
[products=productId,productId2]
[artifacts=fixName1[_version],fixName2[_version]] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	Required. The alias of the target node on which to install fixes.
<i>repo_name</i>	Required. The name of the fix repository from which to install fixes.
[products= <i>productId</i> , <i>productId2</i>]	Optional. The IDs of the products for which you want to install fixes.
[artifacts= <i>fixName1</i> [_version], <i>fixName2</i> [_version]]	Optional. The names of the fixes to install. Specifying the fix version is optional.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- If you do not specify any values for the `artifacts` argument, the command installs the latest version of the official fixes available in a master or image fix repository. If you specify fix names in the `artifacts` argument, the command installs *only*

the specified fixes. If the list of fix names includes engineering fixes, the command installs the specified engineering fixes.

- When installing products on nodes version 9.12 or lower from a mirror repository, if you change the credentials of the Platform Manager on which the mirror repository is stored, you must also update the mirror repository credentials. Use the `sagcc update repository fixes` command to update the mirror repository credentials.

Examples When Executing on Command Central

- To install all fixes in a master fix repository with name “repo1” for all products installed on a target node with alias “sag01”:

```
sagcc exec provisioning fixes sag01 repo1 install
```

- To install all fixes from an image repository with name “MyFixes” for all products installed on a target node with alias “sag01”:

```
sagcc exec provisioning fixes sag01 MyFixes install
```

- To install the fixes with names “wMFix.SPM.TEST1” and “wMFix.SPM.TEST2” and their dependencies from a master repository with name “repo1” on a remote node with alias “sag01”:

```
sagcc exec provisioning fixes sag01 repo1 install
artifacts=wMFix.SPM.TEST1,wMFix.SPM.TEST2
```

- To install all fixes in a fix repository with name “repo1” for the products with IDs “SPM” and “OSGI”:

```
sagcc exec provisioning fixes sag01 repo1 install
products=SPM,OSGI
```

sagcc exec provisioning fixes uninstall

Uninstall a fix. The uninstall operation does not require access to a repository.

Note: A product depends on its Platform Manager plug-in. You cannot uninstall a product without uninstalling the Platform Manager plug-in for that product.

Syntax

- Command Central syntax:

```
sagcc exec provisioning fixes
  node_alias uninstall [artifacts=Id1,Id2]
  [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>node_alias</code>	Required. The alias of the installation from which to uninstall fixes.
<code>[artifacts=Id1, Id2]</code>	Optional. The IDs of the fixes that you want to uninstall. If you omit the artifacts argument, Command Central uninstalls all fixes.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

- When uninstalling a fix, Platform Manager restores the previously installed version of the fix. For example, when you uninstall SPM_9.7_Fix2, the command rolls back SPM_9.7_Fix1.
- If a fix depends on other fixes, uninstalling the fix fails with an error message that lists the fix dependencies. You must include all listed fix dependencies in the `artifacts` argument and run the command again.

Examples When Executing on Command Central

- To uninstall a fix with ID "wMFix.SPM.TEST" from a node with alias "sag01":

```
sagcc exec provisioning fixes sag01 uninstall artifacts=wMFix.SPM.TEST
```

- To uninstall all fixes from a node with alias "sag01":

```
sagcc exec provisioning fixes sag01 uninstall
```

22 Provisioning Products Commands

■ sagcc exec provisioning products install	390
■ sagcc exec provisioning products uninstall	391

sagcc exec provisioning products install

Installs products from a product repository. The Platform Manager on which you install the products must have access to a product repository.

Before installing products using this command, see ["Before Installing Products" on page 41](#).

Syntax

- Command Central syntax:

```
sagcc exec provisioning products node_alias repo_name install
[artifacts=productId1[_version],productId2[_version]] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>node_alias</i>	Required. The alias of the target node on which to install products.
<i>repo_name</i>	Required. The name of the product repository from which to install products.
[artifacts= <i>productId1[_version]</i> , <i>productId2[_version]</i>]	<p>The names of the products to install.</p> <ul style="list-style-type: none"> ■ Required for a master repository. You must specify at least one product with version. ■ Optional for image repositories. If you do not specify any artifacts, Command Central installs all products in the image archive. <p>When you specify more than one product, specifying the version is optional.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p>Note:To find the product IDs, use the sagcc list repository products content command.</p> </div>
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Note

- You cannot install all products available in a master repository. You must specify at least one product with version in the `artifacts` argument.
- When installing products on nodes version 9.12 or lower from a mirror repository, if you change the credentials of the Platform Manager on which the mirror repository is stored, you must also update the mirror repository credentials. Use the `sagcc update repository products` command to update the mirror repository credentials.

Examples When Executing on Command Central

- To install all products from an image repository with name “MyProducts” on a target node with alias “sag01”:

```
sagcc exec provisioning products sag01 MyProducts install
```

- To install the products with product IDs “integrationServer” and “MwsProgramFiles” and their dependencies from a master repository with name “webMethods-EUR” on a remote node with alias “sag01”:

```
sagcc exec provisioning products sag01 webMethods-EUR install
artifacts=integrationServer_9.8,MwsProgramFiles
```

sagcc exec provisioning products uninstall

Uninstall a product. The uninstall operation does not require access to a repository.

Note: A product depends on its Platform Manager plug-in. You cannot uninstall a product without uninstalling the Platform Manager plug-in for that product.

Syntax

- Command Central syntax:

```
sagcc exec provisioning products
  node_alias uninstall [artifacts=Id1, Id2]
  [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>node_alias</code>	Required. The alias of the installation from which to uninstall products.
<code>[artifacts=Id1, Id2]</code>	Optional. The IDs of the products that you want to uninstall. If you omit the artifacts argument,

Argument or Option	Description
	Command Central uninstalls all products, including Platform Manager.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Examples When Executing on Command Central

- To uninstall a product with ID "integrationServer" from a node with alias "sag01":

```
sagcc exec provisioning products sag01 uninstall  
artifacts=integrationServer,ISspm
```

- To uninstall all products, including Platform Manager, from a node with alias "sag01":

```
sagcc exec provisioning products sag01 uninstall
```

23 Repository Commands

■ About Repository Commands	395
■ sagcc add repository assets flatfile	395
■ sagcc add repository assets git	398
■ sagcc delete repository assets	401
■ sagcc list repository assets	401
■ sagcc list repository assets content	402
■ sagcc list repository assets dependencies	403
■ sagcc list repository assets namespaces	404
■ sagcc update repository assets flatfile	405
■ sagcc update repository assets git	407
■ sagcc add repository fixes	409
■ sagcc add repository products	412
■ sagcc delete repository	415
■ sagcc delete repositories	416
■ sagcc exec repository discover	417
■ sagcc exec repository fixes export	418
■ sagcc exec repository refresh	419
■ sagcc list repository	420
■ sagcc list repository discover	421
■ sagcc list repository fixes content	422
■ sagcc list repository fixes dependencies	423
■ sagcc list repository fixes readme	424
■ sagcc list repository products content	425
■ sagcc list repository products dependencies	427
■ sagcc list repository products languages	427
■ sagcc update repository fixes	429

- [sagcc update repository products](#) 431

About Repository Commands

With the repository commands, you can add and manage the following types of source repositories:

Source repository type	Use to...
Product	Install Software AG products
Fix	Install fixes for the products managed by Command Central.
Asset	<p>Deploy assets you create in your development environment to target servers or server groups.</p> <p>An asset repository contains composite assets for Software AG run-time components. The composite assets (or <i>composites</i>) are build using the webMethods Asset Build Environment. For details about composite assets and how to build them, see <i>webMethods Deployer User's Guide</i>.</p>

sagcc add repository assets flatfile

Add a flat file repository that contains user-created source assets.

Syntax

- Command Central syntax:

```
sagcc add repository assets flatfile name=repoName
[location=flat-file-repo-folder-or-zip-on-server|-i flat-file-repo-zip-on-client]
[registryRoot=URL] [registryNamespacesRoot=URL]
[description=description] [overwrite=true|false] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>name=repoName</code>	Required. A unique name for the flat file repository that you are adding.
<code>location=flat-file-repo-folder-or-zip-on-server</code>	The path to the directory on the Command Central server that contains the flat file repository or the zip archive of the flat file.
<code>-i flat-file-repo-zip-on-client</code>	The path to the zip archive of the flat file repository on a client machine.
<code>[registryRoot=URL]</code>	<p>Optional. A valid URL that points to the location of the registry that stores the metadata for the assets in the flat file repository.</p> <p>The registry is located inside the local directory that stores the flat file repository. For example, if the flat file repository is located in the C:\fileRepo directory and the registry directory is "MyRegistry", you must set this parameter as follows: <code>registryRoot=/MyRegistry</code></p>
<code>[registryNamespacesRoot=URL]</code>	<p>Optional. A valid URL that points to the location of the directory that stores the registry namespaces.</p> <p>The namespace directory is located inside the registry directory. For example, if the registry directory is C:\fileRepo, you must set this parameter as follows: <code>registryNamespacesRoot=/MyRegistryNs</code></p>
<code>[description=description]</code>	Optional. A description of the asset repository.
<code>[overwrite=true false]</code>	<p>Optional. Use this argument when re-posting build artifacts from a Jenkins CI job. When the argument is set to <code>true</code>:</p> <ul style="list-style-type: none"> ■ If the repository does not exist, Command Central creates the repository. ■ If the repository exists, Command Central removes the existing repository and creates a new one.

Argument or Option	Description
	The default is <code>true</code> .
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

- You must specify where the flat file repository is located either using the `location` argument, or the `input` file option.
- If you do not specify values for the `registryRoot` and `registryNamespacesRoot` arguments, Command Central resolves the directory path to "." and looks for the registry at the root of:
 - the folder that you specify in the `location` argument
 - in the `Software AG_directory/profiles/CCE/data/flatfile/<repoName>` directory. If you specify a zip archive, Command Central unzips the archive in this directory (regardless of whether the archive is located on the server or client).
- When creating or updating an asset repository, you can check the progress of the operation and troubleshoot issues using "[Jobmanager Jobs Commands](#)" on page or in the Command Central web user interface, go to Stacks > Jobs.

Example When Executing on Command Central

- To add a flat file asset repository with name "f1" from a folder on the Command Central server:

```
sagcc add repository assets flatfile name=f1 location=C:\CommandCentral\AssetRepo
description="Local asset build repository"
```

- To add a flat file asset repository with name "f1" from a zip archive, located on the Command Central server:

```
sagcc add repository assets flatfile name=f1 location=C:\CommandCentral\AssetRepo.zip
description="Local asset build repository"
```

- To add a flat file asset repository with name "f1" from a zip archive, located on the client:

```
sagcc add repository assets flatfile name=f1 -i C:\CommandCentral\AssetRepo.zip
description="Local asset build repository"
```

In this example, `registryRoot` and `registryNamespacesRoot` are not included and Command Central resolves them to the default values:

```
registryRoot="." registryNamespacesRoot="."
```

- To add a flat file asset repository with name "f1" from a zip archive, located on the client, specify "MyRegistry" as the registry directory, and "MyRegistryNs" as the directory for the registry namespaces:

```
sagcc add repository assets flatfile name=f1
-i C:\CommandCentral\AssetRepo.zip registryRoot="/MyRegistry"
registryNamespacesRoot="/MyRegistryNs" description="Local asset build repository."
```

sagcc add repository assets git

Add a remote Git repository that contains user-created source assets.

Syntax

- Command Central syntax:

```
sagcc add repository assets git name=repoName location=URL
[registryRoot=URL] [registryNamespacesRoot=URL] credentials=credAlias
[overwrite=true|false] [description=description] [branch=branch_name] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>name=repoName</code>	Required. A unique name for the remote Git repository that you are adding.
<code>location=URL</code>	Required. A valid URL that points to the location of the remote Git repository. For example, <code>https://github.com/user/repo.git</code>
<code>[registryRoot=URL]</code>	Optional. A valid URL that points to the location of the registry that stores the metadata for the assets in the Git repository. The registry is located inside the local store directory. For example, if the flat file repository is located in the <code>C:\fileRepo</code> directory and the registry directory is "MyRegistry", you must set this parameter as follows: <code>registryRoot=/MyRegistry</code>
<code>[registryNamespacesRoot=URL]</code>	Optional. A valid URL that points to the location of the directory that stores the registry namespaces. The namespace directory is located inside the registry directory. For example, if the registry directory is <code>C:\fileRepo</code> ,

Argument or Option	Description
	<p>you must set this parameter as follows: <code>registryNamespacesRoot=/MyRegistryNs</code></p>
<code>credentials=<i>credAlias</i></code>	<p>Required. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the GIT repository. If the instance already exists, you can retrieve the alias using sagcc list configuration instances. If the configuration instance does not exist, you must create it with the "sagcc create configuration data" on page 213 command. See the "Usage Notes" on page 400 for details about the GIT credentials input properties file that you must provide with the <code>sagcc create configuration data</code> command.</p>
<code>[<i>overwrite=true false</i>]</code>	<p>Optional. Use this argument when re-posting build artifacts from a Jenkins CI job. When the argument is set to <code>true</code>:</p> <ul style="list-style-type: none"> ■ If the repository does not exist, Command Central creates the repository. ■ If the repository exists, Command Central removes the existing repository and creates a new one. <p>The default is <code>true</code>.</p>
<code>[<i>description=description</i>]</code>	<p>Optional. A description of the asset repository.</p>
<code>[<i>branch=branch_name</i>]</code>	<p>Optional. The name of a branch in the git repository from which to pull assets. When you omit this argument, the command uses the branch that is set as default in the remote git repository.</p> <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Note: Use a simple branch name. Do not use prefixes, such as <code>origin/*</code> or <code>refs/*</code></p> </div>
<code>[<i>options</i>]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

- To connect to a GIT repository using certificate authentication over SSH, you must create a COMMON-CREDENTIALS configuration instance using the ["sagcc create configuration data" on page 213](#) command. Specify the following details in an input properties file when running the command:

```
privateKey=URL_to_private_key_file
privateKeyPassword=password_for_private_key_file
knownHosts=URL_to_file_with_known_hosts
```

When you want to use password authentication over HTTP, the credentials input properties file must include the username and password for the user account that can access the GIT repository, for example:

```
username=sag-test
password=the_password_for_sag-test_user
```

For an example of how to create a configuration instance of the COMMON-CREDENTIALS configuration type, see ["COMMON-CREDENTIALS Usage Notes" on page 477](#).

If you are connecting to a public GIT repository that does not require username/password to clone the repository to the local file system, you can specify the default COMMON-CREDENTIALS-NONE configuration instance as the value of the `credentials` argument.

- When creating or updating an asset repository, you can check the progress of the operation and troubleshoot issues using ["Jobmanager Jobs Commands" on page](#) or in the Command Central web user interface, go to Stacks > Jobs.

Examples When Executing on Command Central

- To add a remote GIT repository with name "g1":

```
sagcc add repository assets git name=g1
location=ssh://git@github.com/sag-test/test.git credentials=gitRepoAlias
description="GIT repository hosting the asset build"
```

Command Central connects to the GIT repository over SSH, using the credentials defined for the credentials configuration instance with alias "gitRepoAlias":

```
privateKey=c:/develop-incubator/prototype-repository/build/installs/profiles/
App/workspace/github.priv
privateKeyPassword=manage
knownHosts=c:/develop-incubator/prototype-repository/build/installs/profiles/
App/workspace/known_hosts
```

In this example, `registryRoot` and `registryNamespaceRoot` are not included in the command and Command Central resolves them to the default values:

```
registryRoot="." registryNamespacesRoot="."
```

- To add a remote GIT repository with name "g1" and pull assets from a branch with name "dev":

```
sagcc add repository assets git name=g1
location=ssh://git@github.com/sag-test/test.git credentials=NONE branch=dev
```

sagcc delete repository assets

Deletes asset repositories registered in Command Central.

Syntax

- Command Central syntax:

- To delete all asset repositories:

```
sagcc delete repository assets [options]
```

- To delete a specific repository:

```
sagcc delete repository assets repoName [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repoName</i>	Required. The name of the asset repository that you want to delete.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To delete the asset repository with name "repo1":

```
sagcc delete repository assets repo1
```

sagcc list repository assets

Lists the asset repositories registered in Command Central. Information about the repositories can include:

- Name
- Description, or null if none is assigned
- Repository type
- Location
- Time of last update

Syntax

- Command Central syntax:

```
sagcc list repository assets [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 " Common Options " on page 167.

Usage Notes

Based on the type of the repository, the Last Updated column in the command output shows the time of:

- The last update of a flat-file repository on the file system
- The last commit in a git repository

Examples When Executing on Command Central

To list the registered assets repositories on a Command Central server with host name "rubicon" and port "8490":

```
sagcc list repository assets -server http://rubicon:8490/cce
```

sagcc list repository assets content

Lists the content of an asset repository from all available namespaces.

Syntax

- Command Central syntax:

```
sagcc list repository assets content repoName [type=composite|asset] [options ]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repoName</i>	Required. The name of the asset repository from which to retrieve assets.
[<i>type=composite asset</i>]	<p>Optional. Indicates which type of metadata to retrieve.</p> <ul style="list-style-type: none"> ■ <i>type=composite</i> - list metadata for composite assets ■ <i>type=asset</i> - list metadata only for assets <p>When <i>type</i> is not included, the default is <i>type=composite</i></p>
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

On Linux systems, to display non-ASCII characters correctly in the command results, you must set the encoding of the Linux command-line terminal to UTF-8.

Examples When Executing on Command Central

- To list all composite assets from all namespaces in the repository with name "test_repo":


```
sagcc list repository assets content test_repo
```
- To list assets from all composite assets and all namespaces in the repository with name "test_repo":


```
sagcc list repository assets content test_repo type=asset
```

sagcc list repository assets dependencies

Searches and lists which assets in the repository are required by other assets.

Syntax

- Command Central syntax:

```
sagcc list repository assets dependencies repoName type=asset asset=assetName
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repoName</i>	Required. The name of the asset repository for which you want to list available dependencies.
<i>type=asset</i>	Required. Indicates that you want to retrieve individual assets from the available composites.
<i>asset=assetName</i>	Required. The name of the asset for which you want to list available dependencies.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To list the dependencies for the asset with name "ispackage", included in the repository with name "repo1":

```
sagcc list repository assets dependencies repo1 type=asset asset=ispackage
```

sagcc list repository assets namespaces

Lists all namespaces available in the registry for the specified asset repository.

Syntax

- Command Central syntax:

```
sagcc list repository assets namespaces repoName [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repoName</i>	Required. The name of the asset repository for which you want to list available registry namespaces.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Examples When Executing on Command Central

- To list all available namespaces in the registry for an assets repository with name "test_repo":

```
sagcc list repository assets namespaces test_repo
```

sagcc update repository assets flatfile

Update a flat file repository that contains user-created source assets.

Syntax

- Command Central syntax:

```
sagcc update repository assets flatfile name=repoName
[location=flat-file-repo-folder-or-zip-on-server|-i flat-file-repo-zip-on-client]
[registryRoot=URL] [registryNamespacesRoot=URL]
[description=description] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>name=repoName</i>	Required. A unique name for the flat file repository that you are adding.
<i>location=flat-file-repo-folder-or-zip-on-server</i>	Optional. The path to the directory on the Command Central server that contains the flat file repository or the zip archive of the flat file.

Argument or Option	Description
<code>-i flat-file-repo-zip-on-client</code>	Optional. The path to the zip archive of the flat file repository on a client machine.
<code>[registryRoot=URL]</code>	<p>Optional. A valid URL that points to the location of the registry that stores the metadata for the assets in the repository.</p> <p>The registry is located inside the local directory that stores the flat file repository. For example, if the flat file repository is located in the C:\fileRepo directory and the registry directory is "MyRegistry", you must set this parameter as follows: <code>registryRoot=/MyRegistry</code></p>
<code>[registryNamespacesRoot=URL]</code>	<p>Optional. A valid URL that points to the location of the directory that stores the registry namespaces.</p> <p>The namespace directory is located inside the registry directory. For example, if the registry directory is C:\fileRepo, you must set this parameter as follows: <code>registryNamespacesRoot=/MyRegistryNs</code></p>
<code>[description=description]</code>	Optional. A description of the asset repository.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

- If you do not specify the location of the existing repository, Command Central uses the location of the repository from the cache.
- When creating or updating an asset repository, you can check the progress of the operation and troubleshoot issues using ["Jobmanager Jobs Commands" on page](#) or in the Command Central web user interface, go to Stacks > Jobs.

Example When Executing on Command Central

To update a flat file asset repository with name "repo1" from a zip archive, located on the client, where "MyRegistry" is the name of the registry directory, and "MyRegistryNs" is the name of the directory for the registry namespaces:

```
sagcc update repository assets flatfile name=repo1
```

```
-i C:\CommandCentral\AssetRepo.zip registryRoot="/MyRegistry"
registryNamespacesRoot="/MyRegistryNs" description="Local asset build repository."
```

sagcc update repository assets git

Update a GIT repository that contains user-created source assets.

Syntax

- Command Central syntax:

```
sagcc update repository assets git name=repoName location=URL
[credentials=credAlias] [registryRoot=URL] [registryNamespacesRoot=URL]
[description=description] [branch=branch_name] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>name=<i>repoName</i></code>	Required. A unique name for the flat file repository that you are adding.
<code>[location=<i>URL</i>]</code>	Optional. A valid URL that points to the location of the remote Git repository. For example, https://github.com/user/repo.git
<code>[credentials=<i>credAlias</i>]</code>	Optional. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the GIT repository. If the instance already exists, you can retrieve the alias using sagcc list configuration instances . If the configuration instance does not exist, you must create it with the " sagcc create configuration data " on page 213 command. See the " Usage Notes " on page 400 of the <code>sagcc add repository assets git</code> command for details about the GIT credentials input properties file that you must provide with the <code>sagcc create configuration data</code> command.
<code>[registryRoot=<i>URL</i>]</code>	Optional. A valid URL that points to the location of the registry that stores the metadata for the assets in the repository. The registry is located inside the local directory that stores the flat file repository. For example, if the flat file repository is located in the C:

Argument or Option	Description
	\fileRepo directory and the registry directory is "MyRegistry", you must set this parameter as follows: registryRoot=/MyRegistry
[registryNamespacesRoot= <i>URL</i>]	Optional. A valid URL that points to the location of the directory that stores the registry namespaces. The namespace directory is located inside the registry directory. For example, if the registry directory is C:\fileRepo, you must set this parameter as follows: registryNamespacesRoot=/MyRegistryNs
[description= <i>description</i>]	Optional. A description of the asset repository.
[branch= <i>branch_name</i>]	Optional. The name of a branch in the git repository that you want to update. When you omit this argument, the command uses the branch that is set as default in the remote git repository. Note: Use a simple branch name. Do not use prefixes, such as origin/* or refs/*
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

- If you do not specify the location of the existing repository, Command Central uses the location of the repository from the cache.
- When creating or updating an asset repository, you can check the progress of the operation and troubleshoot issues using "[Jobmanager Jobs Commands](#)" on page or in the Command Central web user interface, go to Stacks > Jobs.

Example When Executing on Command Central

To update a remote GIT repository with name "g1":

```
sagcc update repository assets git name=g1
location=ssh://git@github.com/sag-test/test.git credentials=gitRepoAlias
description="GIT repository hosting the asset build"
```

Command Central connects to the GIT repository over SSH, using the credentials defined for the credentials configuration instance with alias "gitRepoAlias":

```
privateKey=c:/develop-incubator/prototype-repository/build/installs/profiles/
App/workspace/github.priv
privateKeyPassword=manage
knownHosts=c:/develop-incubator/prototype-repository/build/installs/profiles/
App/workspace/known_hosts
```

sagcc add repository fixes

Adds a fix repository in Command Central.

Syntax

- Command Central syntax:

- Master repository:

```
sagcc add repository fixes master name=repo_name location=remote_location
credentials=credAlias [description="description"] [options]
```

- Mirror repository:

```
sagcc add repository fixes mirror name=repo_name
sourceRepos=repo_name1,repo_name2...repo_nameN
[productRepos=repo_name1,repo_name2,...repo_nameN]
[products=product_list] [nodeAlias=node_alias]
[artifacts=fix_list] [description="description"]
```

- Image repository:

In the following command, you must include either the `--input` file option or the `location` argument:

```
sagcc add repository fixes image name=repo_name
[--input | i image_file_on_client | location=image_file_on_server]
[description="description"]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>name=repo_name</code>	Required. The name of the fix repository to add.
<code>[description="description"]</code>	Optional. A description for the repository.
For master repository	
<code>location=remote_location</code>	Required. The URL of the remote master repository.

Argument or Option	Description
<code>credentials=credAlias</code>	Required. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the fix repository. If the instance already exists, you can retrieve the alias using sagcc list configuration instances . If the configuration instance does not exist, you must create it with "sagcc create configuration data" on page 213 .
For mirror repository	
<code>sourceRepos=repo_name1, repo_name2...repo_nameN</code>	Required. A list of the source repositories to include in the new mirror repository.
<code>[productRepos=repo_name1, repo_name2,...repo_nameN]</code>	Optional. Select product repositories for which to add fixes in the fix mirror repository. See the "Usage Notes" on page 411 for some important details about this argument.
<code>[products=productId list]</code>	Optional. A list of IDs of the products for which to include fixes in the mirror repository.
<code>[nodeAlias=node_alias]</code>	Optional. Specifies the alias name of the Platform Manager installation in which you want to add the repository. If you do not include this parameter, the repository is added to the local Platform Manager node.
<code>[artifacts=fix_list]</code>	Optional. A list of the names of the fixes from the source repositories to include in the mirror repository. <code>artifacts=LATEST</code> adds all the latest fixes available in the source repositories.
For image repository	
<code>--input -i image_file_on_client</code>	Required. Identifies an image archive file created with Software AG Installer or Software AG Update Manager. For more information about the <code>--input i</code> option, see "input" on page 177 .
<code>location=image_file_on_server</code>	Required. A valid URL that points to the location where the repository is added. If <code>location</code> points

Argument or Option	Description
	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.
	<p>Important: No repositories can point to the same location.</p>

Usage Notes

When creating a fix mirror repository, Software AG recommends that you specify a list of product repositories or products for which to include fixes, because Empower contains a lot of fixes for different releases and Command Central requires a filter through which to select the fixes.

When creating a fix mirror repository, you must consider the following about the [productRepos] argument:

- You can leave out the list of product repositories in the following cases:
 - When the `sourceRepos` argument lists image repositories. In the following example, the mirror repository is created from a single “isImage” image source repository:

```
sagcc add repository fixes mirror name=Mirror sourceRepos=isImage
```

- When you list specific artifacts in the `artifacts` parameter. In the following example, Command Central will add all versions of the “wmFix.integrationServer.DummyForSPM” fix, specified in the `artifacts` parameter, with all its dependencies:

```
sagcc add repository fixes mirror name=Mirror1 sourceRepos=Mirror2
artifacts=wmFix.integrationServer.DummyForSPM
```

- When you use the `productRepos` argument, listing specific artifacts in the `artifacts` argument is not supported. For example the following command is not supported, because it includes the `productRepos` argument, and the `artifacts` argument points to the specific artifact “wmFix.integrationServer”:

```
sagcc create repository fixes mirror name=myMir sourceRepos=QARepo
productRepos=webMethods-9.12_MirrorRepo
artifacts=wmFix.integrationServer
```

However, when `artifacts=LATEST`, you can use the `productRepos` and `artifacts` arguments in the same command. For example, the following command will create a mirror repository with all the latest fixes for all products installed from the “webMethods-9.12_MirrorRepo” product repository:

```
sagcc create repository fixes mirror name=myMir sourceRepos=QARepo
productRepos=webMethods-9.12_MirrorRepo artifacts=LATEST
```

Examples When Executing on Command Central

- To upload an image file named “image.zip” from the current directory to Command Central and create a fix repository with name “test” that points to that image:

```
sagcc add repository fixes image name=test -i image.zip
```

- To create a mirror repository with name “Mirror1” that contains the latest fixes, selected from Empower, for the products included in the “isImage” and “mwsImage” product image repositories, with all fix dependencies:

```
sagcc create repository fixes mirror name=Mirror1 sourceRepos=Empower
productRepos=isImage,mwsImage artifacts=LATEST
```

- To create a mirror repository with name “Mirror1” that contains fixes selected from Empower for the products installed on the node where the mirror repository is created. You can use this command to create a mirror repository for tested fixes on a specific node.

```
sagcc create repository fixes mirror name=Mirror1 sourceRepos=Empower
products=INSTALLED
```

- To create a mirror repository with name “Mirror1” that contains the latest version of the fixes available on Empower for the product with ID “integrationServer” and version 9.10. The product version is determined from the version of the specified product repository:

```
sagcc add repository fixes mirror name=Mirror1 sourceRepos=Empower
productRepos=webMethods-9.10_CC_PI_TR products=integrationServer
artifacts=LATEST
```

sagcc add repository products

Adds a product repository in Command Central.

Syntax

- Command Central syntax:

- Master repository:

```
sagcc add repository products master name=repo_name
location=remote_location credentials=credAlias
[description="description"]
```

- Mirror repository:

```
sagcc add repository products mirror name=repo_name
sourceRepos=repo_name1,repo_name2...repo_nameN [nodeAlias=node_alias]
[artifacts=productId1,productId2...productIdN]
[platforms=OS_Id1,OS_Id2...OS_IdN] [description="description"]
```

- Image repository:

In the following command, you must include either the `--input file` option or the `location` argument:

```
sagcc add repository products image name=repo_name
[--input | i image_file_on_client | location=image_file_on_server]
```

```
[description="description"]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>name=repo_name</code>	<p>Required. The name of the product repository to add.</p> <p>The name of a mirror repository should follow this format:</p> <pre>webMethods-major.minor_label</pre> <p>The <i>label</i> could be location, platform, or anything meaningful to identify the mirror repository. For example, <code>webMethods-9.9_US_east_linux</code></p>
<code>[description="description"]</code>	Optional. A description for the repository.
For master repository	
<code>location=remote_location</code>	<p>Required. The URL of the remote master repository. To find the URL of the available Software AG product repositories use "sagcc list repository discover" on page 421.</p>
<code>credentials=credAlias</code>	<p>Required. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the product repository. If the instance already exists, you can retrieve the alias using sagcc list configuration instances. If the configuration instance does not exist, you must create it with "sagcc create configuration data" on page 213.</p>
For mirror repository	
<code>[nodeAlias=node_alias]</code>	<p>Optional. Specifies the alias name of the Platform Manager installation in which you want to add the repository. If you do not include this parameter, the repository is added to the local Platform Manager node.</p>
<code>sourceRepos=repo_name1, repo_name2...repo_nameN</code>	<p>Required. A list of the source repositories to include in the new mirror repository.</p>

Argument or Option	Description
<code>[artifacts= productId1,productId2..., productIdN]</code>	<p>Optional. A list of the IDs of the products from the source repositories that you want to include in the mirror repository.</p> <p>If you do not specify a list of product IDs or you omit this parameter, Command Central includes all products from the source repositories.</p>
<code>[platforms= OS_Id1,OS_Id2...OS_IdN]</code>	<p>Optional. A list of the IDs of the operating systems of the products included in the mirror repository.</p> <p>If you do not specify a list of operating system IDs or you omit this parameter, Command Central uses the default platform of the Platform Manager to which the mirror repository is added.</p>
For image repository	
<code>--input - i image_file_on_client</code>	<p>Required. Identifies an image archive file created with Software AG Installer or Software AG Update Manager. For more information about the <code>--input i</code> option, see "input" on page 177.</p>
<code>location= image_file_on_server</code>	<p>Required. A valid URL that points to the location where the repository is added. If <code>location</code> 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.</p>
<p>Important: No repositories can point to the same location.</p>	

Usage Notes

- When you list specific product IDs in the `artifacts` argument, for example `integrationServer`, JDK is not added to the mirror repository, though JDK is required to install Integration Server from the mirror repository. You must add `sjp` in the list of product IDs in the `artifacts` argument, so that the command adds the JDK required for installing the listed products from the mirror repository.
- To add a mirror repository with all products and language packs available in the source repository, you must omit the `artifacts` argument. Command Central does not support listing specific language packs to include in a mirror repository.

Examples When Executing on Command Central

- To find the URLs of the repositories on the server with host "sdc.softwareag.com":

```
sagcc list repository products discover host=sdc.softwareag.com properties=*
```

To add a master product repository, located on the "sdc.softwareag.com" server, using the "myUser" credentials to access the repository:

```
sagcc add repository products master name=webMethods-12
location=http://sdc.softwareag.com/dataservewebM912/repository/
username=myUser password=myUserPassword
```

- 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:

```
sagcc add repository products image name=test -i image.zip
```

- To create a mirror repository with name "webMethods-9.9_EUR_Local" on the local installation for all products available in the "webMethods-9.9_EUR" source repository that use the operating system of the local installation:

```
sagcc create repository products mirror name=webMethods-9.9_EUR_Local
sourceRepos=webMethods-9.9_EUR
```

- To create a mirror repository with name "webMethods-9.9_US_lnxamd64_w64" on a remote installation with alias "repoNode1", from two image repositories with different operating systems:

```
sagcc create repository products mirror name=webMethods-9.9_US_lnxamd64_w64
nodeAlias=repoNode1 sourceRepos=webMethods-9.9_US_lnxamd64,
webMethods-9.9_US_w64
```

- To create a mirror repository with name "webMethods-9.9_US_wMcore" on a remote installation with alias "repoNode2", for two platforms and four core products with all their dependencies:

```
sagcc create repository products mirror name=webMethods-9.9_US_wMcore
nodeAlias=repoNode2 sourceRepos=webMethods-9.9_US
platforms=LNXAMD64,W64
artifacts=integrationServer,NUMRealmServer,TES,MwsProgramFiles
```

sagcc delete repository

Deletes a registered product or fix repository.

Syntax

- Command Central syntax:

```
sagcc delete repository {products | fixes} repo_name
[deleteImage={true | false}] [options]

options:
[--force]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repo name</i>	Required. Specifies the name of the repository that you want to delete.
<code>[deleteImage={true false}]</code>	<p>Optional. Whether to delete the image file referenced by the repository from the file system. Valid values:</p> <ul style="list-style-type: none"> ■ <code>true</code> Delete the image file. ■ <code>false</code> (default) Do not delete the image file. <p>When you do not include this argument, Command Central does not delete the image file from the file system.</p>
<code>[options]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Examples When Executing on Command Central

- To delete a repository with name "repo1" including the image file that the repository references:

```
sagcc delete repository products REPOSITORY-repo1 deleteImage=true
sagcc 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:

```
sagcc delete repository products test
```

sagcc delete repositories

Deletes all registered product or fix repositories.

Syntax

- Command Central syntax:

```
sagcc delete repository {products | fixes}[options]

options:
[--force]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

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 175.

Examples When Executing on Command Central

- To delete all registered product repositories after user confirmation:

```
sagcc delete repository products
```

- To delete all registered fix repositories without user confirmation:

```
sagcc delete repository products --force
```

sagcc exec repository discover

Finds product and fix repositories located on the specified installer server and adds the discovered repositories to Command Central.

Syntax

- Command Central syntax:

```
sagcc exec repository {products|fixes} discover [host=install_server  
[name=sandbox_name]]  
[options]
```

- Platform Manager syntax:

```
sagcc exec repository {products|fixes} discover [host=install_server  
[name=sandbox_name]]  
[options]
```

Arguments and Options

Argument or Option	Description
<code>[host=install_server]</code>	Optional. The host name or IP address of the system hosting the repositories. If you do not

Argument or Option	Description
	specify a value, Command Central goes to the Empower website.
[name= <i>sandbox_name</i>]	Optional. The name of a sandbox on the specified host server. If you do not specify a value, Command Central lists all repositories on the installer server.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

When running the repository discover command, you can specify valid credentials for the repository using the `{--username | -u} user_name` and `{--password | -p} password` options.

Example When Executing on Command Central and Platform Manager

To add a product repository located in the sandbox with name "SuiteProd" from a server with host name "sag":

```
sagcc exec repository products discover host=sag name=SuiteProd
sagcc exec repository fixes discover host=sag name=SuiteProd
```

sagcc exec repository fixes export

Generates a fix image archive from a fix repository.

Syntax

- Command Central syntax:

```
sagcc exec repository fixes export repo_Name
dest=filename.zip artifacts=fixName1 [_version],fixName2 [_version]
[options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. Specifies the name of the repository from which you want to generate a fix image archive.
<code>dest=filename.zip</code>	Required. A name for the generated fix image archive.
<code>artifacts=fixName1[_version], fixName2[_version]</code>	Required. The names of the fixes to include in the image file. Specifying the fix version is optional. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p>Important When a support patch has different versions and you want to export a specific version of the patch, you must specify the fully qualified ID and version of the support patch in the <code>artifacts</code> argument.</p> </div>
<code>platform=OS[_version],</code>	Optional. The ID of the operating system for which to export files.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

To generate a fix image archive with name “myfixes.zip” that contains a fix with name “wMFix.SPM.TEST” from the Empower repository:

```
sagcc exec repository fixes export Empower dest=myfixes.zip
artifacts=wMFix.SPM.TEST
```

sagcc exec repository refresh

Refreshes the products or fixes in a mirror repository.

Syntax

- Command Central syntax:

```
sagcc exec repository {products | fixes} refresh repo_name [options]
```

- Not supported on Platform Manager.

Arguments or Options

Argument or Option	Description
<i>repo_name</i>	Required. The name of the mirror repository that you want to refresh.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

The command uses the source repositories and list or artifacts (products or fixes) defined at the time of creating or updating the mirror repository. When refreshing a fix mirror repository, the command also uses the product repositories and product list, defined at the time of creating or updating the fix mirror repository. For a product mirror repository, the command uses the operating system(s) specified when creating the repository.

Example When Executing on Command Central

- To refresh the product mirror repository with name "webMethods-9.10_myrepo" with the new products from the source repository, from which it was created:

```
sagcc exec repository products refresh webMethods-9.10_myrepo
```

- To refresh the fix mirror repository with name "mirror1" with the new fixes from the source repository, from which it was created:

```
sagcc exec repository fixes refresh mirror1
```

sagcc list repository

Lists registered product or fix repositories.

Syntax

- Command Central syntax:

```
sagcc list repository {products|fixes} [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Examples When Executing on Command Central

To list the registered repositories on a Command Central server with host name "rubicon" and port "8490":

```
sagcc list repository products -server http://rubicon:8490/cce
sagcc list repository fixes -server http://rubicon:8490/cce
```

sagcc list repository discover

Finds product or fix repositories located on the specified installer server, but does not add the discovered repositories to Command Central.

Syntax

- Command Central syntax:

```
sagcc list repository {products|fixes} discover
[host=install_server [name=sandbox_name ] ]
[options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
[<i>host=install_server</i>]	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.
[<i>name=sandbox_name</i>]	Optional. The name of a sandbox on the specified host server. If you do not specify a value, Command Central lists all repositories on the installer server.

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To find a product repository located in the "SuiteProd" sandbox on a server with host name "sag":

```
sagcc list repository products discover host=sag name=SuiteProd
```

To find all product repositories located on a server with host name "sag":

```
cc list repository products discover host=sag properties=*
```

sagcc list repository fixes content

Lists the fixes available in a fix repository.

Syntax

- Command Central syntax:

```
sagcc list repository fixes content repo_name
[products=productId_[version],productId2_[version] | INSTALLED]
[nodeAlias=nodeAlias] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the fix repository for which you want to list available fixes.
<code>[products=productId_[version], productId2_[version] INSTALLED]</code>	Optional. The IDs and version of the products for which to list available fixes. Use commas to separate each product ID. You can determine the IDs for run-time components using the sagcc list inventory components command. When you specify <code>INSTALLED</code> Command Central lists the fixes that match the

Argument or Option	Description
	operating system and installed products on the target installation.
[<i>nodeAlias=nodeAlias</i>]	Optional. Specifies the alias name of a Platform Manager installation. You can determine installation alias names using the sagcc list landscape nodes command.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

When listing the content of a master fix repository, the command returns only the list of official fixes available in the master repository. When listing the content of an image fix repository, the command returns a list of all official and engineering fixes in the repository.

Examples When Executing on Command Central

- To list all available fixes for all products and versions in a fix repository with name "repo1":


```
sagcc list repository fixes content repo1
```
- To list the fixes available for products with IDs "SPM" and "OSGI" for all versions in a repository with name "repo1":


```
sagcc list repository fixes content repo1 products=SPM,OSGI
```
- To list the fixes available for products with IDs "SPM" and "OSGI", and version "9.8" in a fix repository with name "repo1":


```
sagcc list repository fixes content repo1 products=SPM_9.8,OSGI_9.8
```
- To list fixes that match the operating system and installed products on a target node with name "sag01", available in a fix repository with name "repo1":


```
sagcc list repository fixes content repo1 products=INSTALLED
nodeAlias=sag01
```

sagcc list repository fixes dependencies

Checks the dependencies for a fix.

Syntax

- Command Central syntax:

```
sagcc list repository fixes dependencies repo_name fix_name1 [_version],
fix_name2 [_version] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
repo_name	Required. The name of the fix repository that contains the fix.
fix_name1[_version], fix_name2[_version]	Required. The name of the fix or fixes for which you want to check dependencies. Specifying the fix version is optional. Use comma as a list separator.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To check the dependencies for a fix with name "wMFix.SPM.TEST" in a fix repository with name "repo1":

```
sagcc list repository fixes dependencies repo1 wMFix.SPM.TEST
```

sagcc list repository fixes readme

Retrieves the readme for a fix.

Syntax

- Command Central syntax:

```
sagcc list repository fixes readme repo_name fix_name [_version] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the fix repository that contains the fix.
<code>fix_name[_version]</code>	Required. The name of the fix for which you want to retrieve the readme. Specifying the fix version is optional.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example When Executing on Command Central

To retrieve the readme for a fix with name "wMFix.SPM.TEST" in a fix repository with name "repo1":

```
sagcc list repository fixes readme repo1 wMFix.SPM.TEST
```

sagcc list repository products content

Lists the products available in a product repository.

Syntax

- Command Central syntax:

```
sagcc list repository products content repo_name
[products=productId1_[version],productId2_[version] | INSTALLED]
[nodeAlias=nodeAlias] [type=product|langpack|all]
[languages=language1,language2] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the product repository for which you want to list available products.

Argument or Option	Description
<code>[products= productid1_[version], productid2_[version] INSTALLED]</code>	<p>Optional. The IDs and version of the products to list. Use commas to separate each product ID.</p> <p>You can determine the IDs for run-time components using the sagcc list inventory components command.</p> <p>When you specify <code>INSTALLED</code> Command Central lists the products that match the operating system and the products installed on the target installation.</p>
<code>[nodeAlias=nodeAlias]</code>	<p>Required. Specifies the alias name of a Platform Manager installation. You can determine installation alias names using the sagcc list landscape nodes command.</p>
<code>[type=product langpack all]</code>	<p>Optional. Indicates whether to list only products, only language packages, or products and language packages.</p>
<code>[languages=language1,language2]</code>	<p>Optional. Indicates the languages for which to list language packages. Use comma as list separator.</p>
<code>[options]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Examples When Executing on Command Central

- To list all available products for all versions in a product repository with name "repo1":

```
sagcc list repository products content repo1
```

- To list the products with IDs "SPM" and "OSGI", and version "9.8" in a fix repository with name "repo1":

```
sagcc list repository products content repo1  
products=SPM_9.8,OSGI_9.8
```

- To list products that match the operating system and installed products on a target node with name "sag01", available in a product repository with name "repo1":

```
sagcc list repository products content repo1 products=INSTALLED  
nodeAlias=sag01
```

sagcc list repository products dependencies

Checks the dependencies for a product.

Important: Image repositories are not supported by the command.

Syntax

- Command Central syntax:

```
sagcc list repository products dependencies repo_name productId[_version]
[options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<i>repo_name</i>	Required. The name of a master product repository that contains the product.
<i>productId[_version]</i>	Required. The ID of the product for which you want to check dependencies. Specifying the product version is optional.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

To check the dependencies for a product with ID "IS" in a fix repository with name "repo1":

```
sagcc list repository products dependencies repo1 IS
```

sagcc list repository products languages

Lists the languages for which language packages are available in a product repository.

Syntax

- Command Central syntax:

```
sagcc list repository products languages repo_name
[products=productId1[_version],productId2[_version] | INSTALLED]
```

```
[nodeAlias=nodeAlias] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the product repository for which you want to list available languages.
<code>[products=productid1_[version], productid2_[version] INSTALLED]</code>	<p>Optional. The IDs and version of the products for which to retrieve available languages. Use commas to separate each product ID.</p> <p>You can determine the IDs for run-time components using the sagcc list inventory components command.</p> <p>When you specify <code>INSTALLED</code> Command Central lists the products that match the operating system and the products installed on the target installation.</p>
<code>[nodeAlias=nodeAlias]</code>	Required. Specifies the alias name of a Platform Manager installation. You can determine installation alias names using the sagcc list landscape nodes command.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Examples When Executing on Command Central

- To list all available languages for the products in the repository with name "repo1":

```
sagcc list repository products languages repo1
```

- To list the languages available for the products with IDs "SPM" and "OSGI", and version "9.8" in a fix repository with name "repo1":

```
sagcc list repository products languages repo1
products=SPM_9.8,OSGI_9.8
```

- To list the languages for the products that match the operating system and installed products on a target node with name "sag01", available in a product repository with name "repo1":

```
sagcc list repository products languages repo1 products=INSTALLED
nodeAlias=sag01
```

sagcc update repository fixes

Updates fix repository details, such as user credentials, location, and description.

For mirror repositories, the command adds the new fixes specified in the `artifacts` argument to the mirror repository, but does not remove existing fixes or fix versions.

Syntax

- Command Central syntax:

- Master and image repositories:

```
sagcc update repository fixes {master | image} repo_name
[credentials=credAlias]
[location=image_filename | repositoryURL]
[description="description"] [options]
```

- Mirror repositories:

```
sagcc update repository fixes mirror repo_name
[sourceRepos=repo_name1,repo_name2...repo_nameN]
[productRepos=repo_name1,repo_name2,...repo_nameN]
[location=repositoryURL] [products=product_list] [artifacts=fix_list]
[username=username] [password=password]
[description="description"] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the repository that you want to update.
<code>[credentials=credAlias]</code>	Optional. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the fix repository. If the instance already exists, you can retrieve the alias using sagcc list configuration instances . If the configuration instance does not exist, you must create it with " sagcc create configuration data " on page 213.
<code>[location=image_file repositoryURL]</code>	Optional. The name of the image file or valid URL to the new location

Argument or Option	Description
	of the image file referenced by the repository. For mirror repositories, the only valid value is <code>[location=repositoryURL]</code>
<code>[sourceRepos=repo_name1, repo_name2...repo_nameN]</code>	Optional. A list of the source repositories from which to update the mirror repository.
<code>[productRepos=repo_name1, repo_name2...repo_nameN]</code>	Optional. A list of product repositories for which to add fixes in the fix mirror repository.
<code>[products=product_list productId_version_list INSTALLED]]</code>	Optional. A list of IDs of the products for which to update the fixes in the mirror repository.
<code>[artifacts=fix_list fixId_version_list LATEST ALL]</code>	Optional. A list of the names of the fixes from the source repositories that you want to update in the mirror repository.
<code>[description="description"]</code>	Optional. The new description of the repository.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

When installing products on nodes version 9.12 or lower from a mirror repository, if you change the credentials of the Platform Manager on which the mirror repository is stored, you must also update the mirror repository credentials. Use the `sagcc update repository fixes` command to update the mirror repository credentials.

Examples When Executing on Command Central

- To update the user name and password for a fix repository with name "QARepo":

```
sagcc update repository fixes mirror QARepo username=sum password=secret
```

- To change the location of the file “image.zip” referenced by the fix repository with name “repo1”:

```
sagcc update repository fixes image repo1 location=file:///vmtest/image.zip
```

sagcc update repository products

Updates product repository details, such as user credentials, location, and description.

For mirror repositories, the command adds the new products specified in the `artifacts` argument to the mirror repository, but does not remove existing products or product versions.

Syntax

- Command Central syntax:

- Master and image repositories:

```
sagcc update repository products {master | image} repo_name
[credentials=credAlias]
[location=image_filename | repositoryURL]
[description="description"] [options]
```

- Mirror repositories:

```
sagcc update repository products mirror repo_name
[sourceRepos=repo_name1,repo_name2...repo_nameN]
[location=repositoryURL]
[artifacts=productId1,productId2...productIdN]
[platforms=OS_Id1,OS_Id2...OS_IdN]
[username=username] [password=password]
[description="description"] [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
<code>repo_name</code>	Required. The name of the repository that you want to update.
<code>--input i filename.xml</code>	Required. An XML file that contains repository data, such as repository name, description, and location.
<code>[credentials=credAlias]</code>	Optional. The alias of the COMMON-CREDENTIALS configuration instance that has the details for accessing the product repository. If the instance already exists, you can retrieve the alias using

Argument or Option	Description
	<p>sagcc list configuration instances. If the configuration instance does not exist, you must create it with "sagcc create configuration data" on page 213.</p>
<p><code>[location=<i>image_file</i> <i>repositoryURL</i>]</code></p>	<p>Optional. The name of the image file or valid URL to the new location of the image file referenced by the repository.</p> <p>For mirror repositories, the only valid value is <code>[location=<i>repositoryURL</i>]</code></p>
<p><code>[sourceRepos=<i>repo_name1</i>, <i>repo_name2</i>...<i>repo_nameN</i>]</code></p>	<p>Optional. A list of the source repositories from which to update the mirror repository.</p>
<p><code>[artifacts= <i>productId1</i>,<i>productId2</i>..., <i>productIdN</i>]</code></p>	<p>Optional. A list of IDs of the products from the source repositories that you want to update in the mirror repository.</p>
<p><code>[platforms= <i>OS_Id1</i>,<i>OS_Id2</i>...<i>OS_IdN</i>]</code></p>	<p>Optional. A list of the IDs of the operating systems of the products included in the mirror repository.</p> <p>If you do not specify a list of operating system IDs or you omit this parameter, Command Central uses the default platform of the Platform Manager to which the mirror repository is added.</p>
<p><code>[description="<i>description</i>"]</code></p>	<p>Optional. The new description of the repository.</p>
<p><code>[options]</code></p>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167.</p>

Usage Notes

When installing products on nodes version 9.12 or lower from a mirror repository, if you change the credentials of the Platform Manager on which the mirror repository is stored, you must also update the mirror repository credentials. Use the `sagcc update repository products` command to update the mirror repository credentials.

Example When Executing on Command Central

- To change the location of the file “image.zip” referenced by the repository with name “repo1”:

```
sagcc update repository products image repo1 location=file:///vmtest/image.zip
```

- To update the local mirror repository with name “webMethods-9.9_EUR_Local” with the five new products listed in the `artifacts` argument for three different platforms:

```
sagcc update repository products mirror webMethods-9.9_EUR_Local  
platforms=LNAMD64,W64,OSX artifacts=DatabaseComponentConfigurator,  
integrationServer,NUMRealmServer,TES,MwsProgramFiles
```

- To update the local mirror repository with name “webMethods-9.9_US_Local” with all artifacts from the source repositories associated with this mirror repository, after creating or modifying the repository:

```
sagcc update repository products mirror webMethods-9.9_US_Local artifacts=all
```

- To update the user name and password for the local mirror repository with name “webMethods-9.12_EUR_local”:

```
sagcc update repository products mirror webMethods-9.12_EUR_local username=sum  
password=secret
```


24 Resources Commands

- sagcc list resources icons 436

sagcc list resources icons

Lists information about the installed icons. This command is useful if you want to use the [sagcc 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:

```
sagcc 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
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see "Common Options" on page 167 .

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:

```
sagcc 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" on page 192](#). The command specifies "secret" for the user's password.

25 Security Credentials Commands

■ sagcc add security credentials	438
■ sagcc add security credentials sharedsecret	441
■ sagcc delete security credentials	443
■ sagcc get security credentials	444

sagcc add security credentials

Adds security credentials for an installation or run-time component. Command Central does not store security credentials for run-time components. Command Central only stores the credentials it uses to connect to Platform Manager nodes. When adding security credentials for all run-time components, Command Central propagates the credentials to the run-time component installation. Command Central does not propagate *only* the credentials for the connection between Command Central and the Platform Manager installation.

Syntax

- Command Central syntax:

- With parameters

```
sagcc add security credentials nodeAlias=node_alias
[runtimeComponentId=componentid]
authenticationType={NONE|BASIC|TRUSTED} [username=username]
[password=password] [options]
```

- With input file

```
sagcc 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= <i>node_alias</i>	<p>Required. Specifies the alias name of the installation for which you want to associate the security credentials.</p> <p>You can view a list of installations and their aliases using sagcc list landscape nodes.</p>

Argument or Option	Description
<code>[runtimeComponentId=<i>componentid</i>]</code>	<p>Optional. Specifies the run-time component for which you want to associate the security credentials.</p> <p>You can determine the IDs for run-time components using sagcc list inventory components.</p> <div data-bbox="850 562 1370 835" style="background-color: #f0f0f0; padding: 5px;"> <p>Important: When configuring the credentials used for communication between Command Central and the Platform Manager installation, set this argument as follows: <code>runtimeComponentId=SPM-CONNECTION.</code></p> </div>
<code>authenticationType={NONE BASIC TRUSTED}</code>	<p>Required when you use the command syntax with parameters. Specifies the authentication type to use for the run-time component. Valid values:</p> <ul style="list-style-type: none"> ■ NONE ■ BASIC - use basic authentication ■ TRUSTED - use trusted authentication
<code>[username=<i>username</i>]</code>	<p>Optional. Specifies the user name to use when the authentication type is set to BASIC.</p>
<code>[password=<i>password</i>]</code>	<p>Optional. Specifies the password to use when the authentication type is set to BASIC.</p>
<code>{--input -i} <i>file</i>{.xml .json}</code>	<p>Required when you use the command syntax with input file. Identifies an input file that contains the data for the security credentials. For more information, see "input" on page 177.</p> <div data-bbox="850 1793 1370 1911" style="background-color: #f0f0f0; padding: 5px;"> <p>Tip: To determine how to specify the data in the input file, use sagcc get security credentials to retrieve data</p> </div>

Argument or Option	Description
	for existing security credentials. For example, if you want to use an XML input file, use <code>sagcc get security credentials</code> with the <code>--format xml</code> option to retrieve the data in XML format.
<code>[options]</code>	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see " Common Options " on page 167.

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:

```
sagcc add security credentials runtimeComponentId=SPM-CONNECTION
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](#)" on page 189 and "[username](#)" on page 192. The commands specify "secret" for the user's password.

- To update the security credentials used for the communication between Command Central and the Platform Manager installed in the installation with alias name "sag01", using authentication type "BASIC":

```
sagcc add security credentials nodeAlias=sag01
runtimeComponentId=SPM-CONNECTION
authenticationType=BASIC username=Administrator password=secret
```

- To update the security credentials used for the communication between Command Central and the Platform Manager installed in the installation with alias name "sag01", using an input file:

- After changing the Command Central default administrator password, create the `c:\inputs\credentials_osgi.xml` file. Open the `credentials_osgi.xml` file and include the updated security credentials for the run-time component with ID “SPM-CONNECTION”.

- Execute the following command:

```
sagcc add security credentials nodeAlias=sag01
runtimeComponentId=SPM-CONNECTION --input credentials_osgi.xml
-p newpassword
```

- To update the security credentials for a run-time component with ID “MwsProgramFiles” installed in the installation with alias name “sag01”, using trusted authentication:

```
sagcc add security credentials nodeAlias=sag01
runtimeComponentId=MwsProgramFiles
authenticationType=TRUSTED
```

- 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:

```
sagcc 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:

```
sagcc add security credentials nodeAlias=sag01
runtimeComponentId=IntegrationServer_instanceName
--input c:\inputs\creds_data.xml --password secret
sagcc add security credentials nodeAlias=sag02
runtimeComponentId=IntegrationServer_instanceName
--input c:\inputs\creds_data.xml --password secret
```

sagcc add security credentials sharedsecret

Add shared secret for a landscape or an environment.

Syntax

- Command Central syntax:

```
sagcc add security credentials sharedsecret [environmentAlias=alias]
secret=text_string [options]
```

- Not supported by Platform Manager.

Arguments and Options

Argument or Option	Description
[environmentAlias= <i>alias</i>]	Optional. Specify the alias for the environment for which you want to set a shared secret.

Argument or Option	Description
	If you omit this parameter, the command sets a global shared secret for the whole landscape.
<code>secret=text_string</code>	Specify a shared secret password that Platform Manager uses to encrypt or decrypt configuration passwords.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Examples When Executing on Command Central

- To configure “mysecret” as the global shared secret for the landscape:

```
sagcc add security credentials sharedsecret secret=mysecret
```

- To configure “mysecret123” as the shared secret for the environment with alias “env1”:

```
sagcc add security credentials sharedsecret environmentAlias=env1
secret=mysecret123
```

- To update the proxy server details for a run-time component with ID “OSGI-SPM” that runs in the installation with alias name “sag01” that belongs to an environment with alias “Test”:

1. Configure “secret123” as the shared secret for the “Test” environment”:

```
sagcc add security credentials sharedsecret
environment=Test secret=secret123
```

2. Retrieve the instance data for the configuration instance with ID “COMMON-PROXY” for the “OSGI-SPM” run-time component that runs in the “sag01” installation:

```
sagcc get configuration data sag01 OSGI-SPM COMMON-PROXY
--output configWithEncryptedData.xml
```

The `configWithEncryptedData.xml` file contains the proxy details including the password, encrypted with the “secret123” shared secret configured for the “Test” environment.

3. Edit the proxy details in the `configWithEncryptedData.xml` file as required without changing the encrypted password.
4. Update the instance data for the configuration instance with ID “COMMON-PROXY”:

```
sagcc update configuration data sag01 OSGI-SPM COMMON-PROXY
--input configWithEncryptedData.xml
```

Command Central updates the proxy details using the data in the `configEncryptedData.xml` file and the “secret123” shared secret configured for the environment.

sagcc delete security credentials

Deletes security credentials from an installation or run-time component.

Syntax

- Command Central syntax:

```
sagcc 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= <i>node_alias</i>]	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 sagcc list landscape nodes .
[runtimeComponentId= <i>componentid</i>]	Optional. Specifies the run-time component for which you want to delete security credentials. You can determine the IDs for run-time components using sagcc list inventory components .
[<i>options</i>]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the

Argument or Option	Description
	options, see "Common Options" on page 167 .

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":

```
sagcc 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":

```
sagcc delete security credentials
runtimeComponentId=IntegrationServer-instanceName
--username Administrator --password manage
```

sagcc get security credentials

Retrieves security credentials that are associated with an installation or run-time component.

Syntax

- Command Central syntax:

```
sagcc 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 sagcc 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 sagcc 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 "Common Options" on page 167 .

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:

```
sagcc get security credentials nodeAlias=sag01
runtimeComponentId=IntegrationServer-instanceName --format xml
```

```
--server http://rubicon:8090/cce --username Administrator  
--password manage
```

26 Stacks and Layers Commands

■ sagcc create stacks	448
■ sagcc create stacks layers	449
■ sagcc create stacks layers nodes master	450
■ sagcc delete stacks	451
■ sagcc delete stacks layers	452
■ sagcc list stacks	453
■ sagcc list stacks layers	454
■ sagcc list stacks layers fixes	454
■ sagcc list stacks layers fixes compare	455
■ sagcc list stacks layers instances	456
■ sagcc list stacks layers instances compare	457
■ sagcc list stacks layers nodes	458
■ sagcc get stacks layers nodes master	459
■ sagcc list stacks layers products	460
■ sagcc list stacks layers products compare	461

sagcc create stacks

Creates a new product stack.

Syntax

- Command Central syntax:

```
sagcc create stacks alias=stackName release=version
[description=description] [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<code>alias=<i>stackName</i></code>	Required. The name of the stack. Specify a name that is unique among all stacks this Command Central manages.
<code>release=<i>version</i></code>	Required. The release version of the stack. For example, 10.1
<code>description=<i>description</i></code>	Optional. A description of the new stack. If you type a description with spaces, place quotes around the description.
<code>[<i>options</i>]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

After creating a new stack, use the `sagcc create stacks layers` command to add infrastructure and run-time layers to the stack. A stack should typically have at least one infrastructure and one run-time layer.

Example

To create a new stack with alias "stack01", release version "10.1" and description "test environment stack":

```
sagcc create stacks alias=stack01 release=10.1 description="test environment stack"
--password secret
```

Because the `--server | -s` and `--username | -u` options are not specified, the command uses the default server and user name. The command uses "secret" as the password for the user.

sagcc create stacks layers

Creates a layer in an existing stack.

Syntax

- Command Central syntax:

```
sagcc create stacks stackName layers alias=layerName layerType=type
nodes=node_aliases [repoProduct=repoName repoFix=repoName
repoAsset=repoName] [templateProperty1=templateValue1 ...]
[description=description] [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack in which to add the layer.
alias= <i>layerName</i>	Required. The name of the layer. Specify a name that is unique among the layers in the stack.
layerType= <i>layerTypeId</i>	Required. The alias of a configuration instance of a layer definition, for example <code>INFRA-LOCAL</code> is the alias of the instance with ID <code>CCE-LAYER-TYPES-infra-local</code> . Go to Command Central Server > Configuration > Layer Types in the Command Central web user interface or use the CLI configuration commands to create configuration instances of layer definitions.
nodes= <i>node_aliases</i>	Required. A comma separated list of the Platform Manager nodes to include in the layer.
[<i>repoProduct=repoName</i> <i>repoFix=repoName</i> <i>repoAsset=repoName</i>]	Optional. The name of a product, fix, and asset repository to include in the layer. When creating an infrastructure layer, you can use the <code>cc.installer=repoName</code> parameter to point

Argument or Option	Description
	to the bootstrap installer from which to provision Platform Manager
<code>[templateProperty1=<i>templateValue</i>]</code>	Optional. User-defined parameters for the micro template referenced in the layer definition.
<code>description=<i>description</i></code>	Optional. A description of the new stack. If you type a description with spaces, place quotes around the description.
<code>[<i>options</i>]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example

To create a new layer with name "layer01" in the stack with alias "stack01", using the layer definition with alias "INFRA-CONNECT", and include the nodes with alias "sag01" and "sag02" in the layer:

```
sagcc create stacks stack01 layers alias=layer01 layerType=INFRA-CONNECT
nodes=sag01,sag02 --password secret
```

The command uses "secret" as the password for the user.

sagcc create stacks layers nodes master

Defines the master node in a layer. When you create a layer, Command Central sets a default master node to which it compares all other nodes defined for the layer. Use this command only if you want to assign as master another node instead of the default master node.

Syntax

- Command Central syntax:

```
sagcc create stacks stackName layers layerName
nodes master node_alias [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack to which the layer belongs.
<i>layerName</i>	Required. The name of the layer for which to define a master node.
<i>node_aliases</i>	Required. The alias of the Platform Manager node to set as the master node in the layer.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example

To define the Platform Manager node with alias "sag01" as the master node for the layer with alias "layer01" in the stack with alias "stack01":

```
sagcc create stacks stack01 layers layer01 nodes master sag01 --password secret
```

The command uses "secret" as the password for the user.

sagcc delete stacks

Deletes a stack. Before deleting the stack, you must delete all layers in the stack.

Syntax

- Command Central syntax:

```
sagcc delete stacks stackName [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack that you want to delete.

Argument or Option	Description
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

The `sagcc delete stacks` command does not physically delete the stack(s) and does not uninstall the products and fixes in the stack(s). It just removes the stack(s) from Command Central management.

Example

To delete a stack with name "layer01":

```
sagcc delete stacks stack01
```

sagcc delete stacks layers

Deletes a layer from a stack.

Syntax

- Command Central syntax:

```
sagcc delete stacks stackName layers layerName [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack to which the layer that you want to delete belongs.
<i>layerName</i>	Required. The name of the layer that you want to delete.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

The `sagcc delete stacks layers` command does not physically delete the layer(s) and does not uninstall the products and fixes in the layer(s). It just removes the layer(s) from the stack.

Example

To delete a layer with name "layer01" in the stack with alias "stack01":

```
sagcc delete stacks stack01 layers layer01
```

sagcc list stacks

Lists all stacks that Command Central manages. Information about the stacks can include:

- Alias name
- Description, or null if none is assigned
- Release version

Syntax

- Command Central syntax:

```
sagcc list stacks [options]
```

- Not applicable to 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 "Common Options" on page 167 .

Example

To list all stacks managed by the Command Central server with host name "rubicon" and port "8490", using the credentials of the Administrator user:

```
sagcc list stacks --format xml --server http://rubicon:8490/cce
--username Administrator --password manage
```

The example command returns the information in XML format.

sagcc list stacks layers

Lists all layers in a stack. Information about the layers can include:

- Alias name
- Description, or null if none is assigned
- Type

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example

To list all layers included in the stack with name "stack01", using the password of the default user:

```
sagcc list stacks stack01 layers --format xml --password manage
```

The example command returns the information in XML format.

sagcc list stacks layers fixes

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

- Fix ID
- Fix name
- Version of the fix

- Product to which the fix is applied
- The date and time when the fix was installed

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName fixes [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example

To list all fixes applied to products in the layer with name "layer01" in the stack with alias "stack01" that the Command Central with host name "rubicon" and port "8090" manages, using the authorization of the default user, and have the information returned to the output file "fixlist" in XML format:

```
sagcc list stacks stack01 layers layer01 fixes --format xml --output fixlist
--server http://rubicon:8090/cce --username Administrator --password manage
```

sagcc list stacks layers fixes compare

Compare information about fixes that have been applied to products in a layer.

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName
fixes compare [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example

To compare all fixes applied to products in the layer with name "layer01" in the stack with alias "stack01" that the Command Central with host name "rubicon" and port "8090" manages, using the authorization of the default user, and have the information returned in XML format:

```
sagcc get stacks stack01 layers layer01 fixes compare --format xml
--server http://rubicon:8090/cce --username Administrator --password manage
```

sagcc list stacks layers instances

Lists all run-time instances that belong to a stack layer. Information about the instances can include:

- Node alias
- Run-time component ID
- Display name
- Category
- Product ID

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName instances [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example

To list all run-time instances included in the layer with name "layer01" in the stack with name "stack01", managed by the Command Central server with host name "rubicon" and port "8090", using the password of the default user:

```
sagcc list stacks stack01 layers layer01 instances --format xml
--server http://rubicon:8090/cce --username Administrator --password manage
```

The example command returns the information in XML format.

sagcc list stacks layers instances compare

Compares the run-time instances that belong to a stack layer.

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName
instances compare [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.

Argument or Option	Description
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

The command returns the results of the comparison. Each row in the results shows a run-time instance property and information about whether this property matches on all nodes of the layer.

Example

To compare the properties of all run-time components included in the layer with name "layer01" in the stack with name "stack01", managed by the Command Central server with host name "rubicon" and port "8090", using the password of the default user:

```
sagcc list stacks stack01 layers layer01 instances compare --format xml
--server http://rubicon:8090/cce --username Administrator --password manage
```

The example command returns the information in XML format.

sagcc list stacks layers nodes

Lists the installations that belong to a layer. Information about the installations can include:

- Alias name
- Release version
- URL of the Platform Manager that manages the installation

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName nodes [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Notes

To see all details about a layer, run the command with the `properties=*` argument.

Example

To list all installations included in the layer with name "layer01" in the stack with name "stack01", managed by the Command Central with host name "rubicon" and port "8090", using the credentials of the default user, and have the information returned to the output file "nodelist" in XML format:

```
sagcc list stacks stack01 layers layer01 nodes --format xml --output nodelist
--server http://rubicon:8090/cce --username Administrator --password manage
```

sagcc get stacks layers nodes master

Retrieve the master installation in a stack layer.

- Alias name
- Release version
- URL of the Platform Manager that manages the installation

Syntax

- Command Central syntax:

```
sagcc get stacks stackName layers layerName nodes master [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

To see all details about a master node, run the command with the `properties=*` argument.

Example

To retrieve the master node for the layer with alias "layer01" in the stack with alias "stack01":

```
sagcc get stacks stack01 layers layer01 nodes master --password secret
```

The command uses "secret" as the password for the default user.

sagcc list stacks layers products

Lists the products included in a stack layer. Information about the products can include:

- Node alias
- Product ID
- Product display name
- Product version

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName products [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.
<i>layerName</i>	Required. The name of the layer for which you want to get information.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Example

To list all products included in the layer with name "layer01" in the stack with alias "stack01" that the Command Central with host name "rubicon" and port "8090" manages, using the authorization of the default user, and have the information returned to the output file "productlist" in XML format:

```
sagcc list stacks stack01 layers layer01 products --format xml --output productlist
--server http://rubicon:8090/cce --username Administrator --password manage
```

sagcc list stacks layers products compare

Compares the products that belong to a layer.

Syntax

- Command Central syntax:

```
sagcc list stacks stackName layers layerName
products compare [options]
```

- Not applicable to Platform Manager.

Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which you want to get information.

Argument or Option	Description
<i>layerName</i>	Required. The name of the layer for which you want to get information.
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Usage Notes

The command returns the results of the comparison. The results show whether the release versions of the products in the layer match and also list the release version for each node alias.

Example

To compare all products included in the layer with name "layer01" in the stack with name "stack01", managed by the Command Central server with host name "rubicon" and port "8090", using the password of the default user:

```
sagcc get stacks stack01 layers layer01 products compare --format xml  
--server http://rubicon:8090/cce --username Administrator --password manage
```

The example command returns the information in XML format.

27

Template Commands

■ sagcc delete templates composite	464
■ sagcc exec templates composite apply	464
■ sagcc exec templates composite import	467
■ sagcc exec templates composite validate	468
■ sagcc get templates composite export	469
■ sagcc list templates composite	470

sagcc delete templates composite

Deletes a composite template with the specified alias.

Syntax

- Command Central syntax:

```
sagcc delete templates composite template_alias [options]
  options :
  [--force]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>template_alias</i>	Required. The alias of the template to delete. You can determine the template alias using the <code>sagcc get templates composite</code> command.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

To delete a template with name “testTemplate”:

```
sagcc delete templates composite testTemplate -p mypassword
```

sagcc exec templates composite apply

Applies a composite template registered and available under the specified alias in a Command Central installation.

Syntax

- Command Central syntax:

- With input parameters:

```
sagcc exec templates composite apply template_alias
environment.mode={provision | migration | migrate}
param1=v1 param2=v2 [param3=v3...paramn=vn] [options]
```

- With input properties file:

```
sagcc exec templates composite apply template_alias
```

```
{--input | -i} filename.properties [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>template_alias</code>	Required. The alias for the template to apply. You can determine the template alias using the <code>sagcc list templates composite</code> command.
<code>environment.mode={provision migrate migration}</code>	Optional for the provisioning operation, but required when migrating an environment. Indicates the type of operation the apply command will execute. Valid values: <ul style="list-style-type: none"> ■ <code>provision</code> - the command installs a new environment ■ <code>migrate migration</code> - the command migrates an existing environment
<code>param1=v1 param2=v2 [param3=v3...paramn=vn]</code>	Required. The input parameters declared in the <code>template_alias</code> YAML file.
<code>{--input -i} filename.properties</code>	Required. A properties file that contains the input parameters declared in the <code>template_alias</code> YAML file
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see " Common Options " on page 167.

Usage Note

- You can include both the `param1=v1 param2=v2 [param3=v3...paramn=vn]` argument and an input properties file in the same command. The values of the input parameters from the argument will override the values of the same parameters specified in the properties file.
- When applying a composite template on the local installation, Command Central and Platform Manager will restart and the apply composite template job does not show on the jobs list.
- If one of the target nodes defined in a composite template is the localhost node, Command Central does not get restarted during the composite template application on the local node.

- When you include the `environment.mode=migrate` parameter, the `apply` command migrates the environment, but does not create new product instances. The product instances get created by the product migration utilities as described in *Upgrading Software AG Products*.
- When applying a composite template, the main composite template `apply` job returns DONE status even when a sub-job in the composite template that Command Central applies fails. However, the main composite template `apply` job also returns `customStatus=WARNING` to indicate that a sub-job has failed.

Examples When Executing on Command Central

- The composite template with alias “myTemplate” is already imported in a Command Central installation. The template provisions an environment with type “dev” and installs a Platform Manager with alias “dev2” and port “7292” in the specified installation directory:

```
sagcc exec templates composite apply myTemplate environment.type=dev
spm.alias=dev2 spm.port=7292 install.dir=C:\dev\nodes\dev2
```

- To apply a template with alias “myTemplate”, which is already imported in a Command Central installation, using the input parameters specified in the “mydevenv.properties” file:

```
sagcc exec templates composite apply myTemplate -i mydevenv.properties
```

- The composite template with alias “myTemplate” does not include a migration section and uses default migration settings. The template is applied on the “is1” and “is3” hosts:

```
sagcc exec templates composite apply myTemplate environment.mode=migrate
hosts=[is1,is3]
```

Because “myTemplate” uses default migration settings, the `apply` command renames the source installation directory and applies the source ports and database defined in the template to the target environment.

- The “myTemplate” composite template uses default migration settings and is applied on the “is1” and “is3” hosts. The `apply` command migrates “myTemplate” from the “/opt/sag98” source directory to the “/opt/sag910” target installation directory:

```
sagcc exec templates composite apply myTemplate environment.mode=migrate
hosts=[is1,is3] src.install.dir=/opt/sag98 install.dir=/opt/sag910
```

- The command applies a template with alias “myTemplate”, which is already imported in a Command Central installation, using the input parameters specified in the “env.properties” file. The command waits for Command Central to become online for 120 seconds and submits the `apply` template request, then starts monitoring the job. If Command Central gets restarted at some point in the `apply` template operation, the command waits for Command Central to come online for 120 seconds. If Command Central does not come online for 120 seconds, the job fails. If Command Central does come online within 120 seconds, the command is resubmitted. The command fails if it is not executed within 3600 seconds, or 1 hour.

```
sagcc exec templates composite apply myTemplate --wait-for-cc 120 --sync-job
```

```
-w 3600 -e "DONE" -c 30 --retry 1 -i env.properties
```

sagcc exec templates composite import

Registers an existing composite template in a Command Central installation. With this command, you can import a composite template that is:

- a single YAML template definition file
- a zip archive that contains the YAML template definition and other files

Syntax

- Command Central syntax:

```
sagcc exec templates composite import {--input | -i} filename.{zip| yaml}
overwrite={true | false} [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
{--input -i} filename.{zip yaml}	Required. An input file that contains a composite template. For more information about this command option, see "input" on page 177 .
overwrite={true false}	Required when you execute the command to update an environment, using an updated version of the same composite template. Specifies whether to delete a composite template that is already imported in Command Central. Valid values: <ul style="list-style-type: none"> ■ true - delete the template ■ false - (default) do not delete the template
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

- To import a composite template from the "myTemplate.zip" file into a Command Central installation:

```
sagcc exec templates composite import --input c:\tmp\myTemplate.zip
--input-format application/zip
```

- To delete an imported composite template:

```
sagcc exec templates composite import -i template.yaml overwrite=true
```

sagcc exec templates composite validate

Validates the composite template with the specified alias.

Syntax

- Command Central syntax:

```
sagcc exec templates composite validate template_alias  
[templateParameter=value...] [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<i>template_alias</i>	Required. The alias of the template to validate. You can determine the template alias using the <code>sagcc get templates composite</code> command.
[<i>templateParameter=value...</i>]	Optional. Specify template parameters that determine which validation checks the command will perform.
[<i>options</i>]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Skipping Validation Checks

You can skip some parts of the template validation by setting the template parameters in the following table to `true`:

Argument or Option	Description
<code>skip.runtime.validation=true</code>	Does not validate the entire template.
<code>skip.repo.conn.validation=true</code>	Does not check the connection to the repositories and if the products and fixes defined in the template and their

Argument or Option	Description
	dependencies are available in the repositories.
<code>skip.product.dep.validation=true</code>	Does not check if the products and their dependencies are available in the product repositories.
<code>skip.fix.dep.validation=true</code>	Does not check if the fixes and their dependencies are available in the fix repositories.

Usage Notes

The checks of the template validity that the command performs are described in the ["Validating a Composite Template" on page 114](#) topic.

Example When Executing on Command Central

- To validate a template with alias name "testTemplate":

```
sagcc exec templates composite validate testTemplate -p mypassword
```

- To skip checking if the fixes defined in the template with alias name "testTemplate" and their dependencies are available in the fix repositories:

```
sagcc exec templates composite validate testTemplate skip.fix.dep.validation=true
```

sagcc get templates composite export

Exports the composite template available under the specified alias into a zip archive.

Syntax

- Command Central syntax:

```
sagcc get templates composite export template_alias
{--output | -o} filename.zip [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code><i>template_alias</i></code>	Required. The alias of the template to export. You can determine the template alias using the <code>sagcc list templates composite</code> command.

Argument or Option	Description
<code>{--output -o}</code> <code>filename.zip</code>	Required. The name of the output zip file to which to export the template. For more information about the <code>{--output -o}</code> option, see "output" on page 182 .
<code>[options]</code>	The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

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":

```
sagcc get templates composite export myAlias
--server http://rubicon:8090/cce -p mypassword
--output template-output-file.zip
```

sagcc list templates composite

Retrieves a list of composite templates available in a landscape.

Syntax

- Command Central syntax:

```
sagcc list templates composite [options]
```

- Not supported on Platform Manager.

Arguments and Options

Argument or Option	Description
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see "Common Options" on page 167 .

Example When Executing on Command Central

To list the composite templates on the Command Central server with host name "rubicon" and port "8090":

```
sagcc list templates composite --server http://rubicon:8090/cce -p mypassword
```

28

Configuring Command Central and Platform Manager

Manager

■ Configuration Types for Command Central and Platform Manager OSGI	472
■ Configuration Types for Command Central and Platform Manager OSGI ENGINE	475
■ Run-time Monitoring States for OSGI-CCE and OSGI-SPM	479
■ Run-time Monitoring States for OSGI-SPM-ENGINE	480
■ Server System Properties	481
■ Prerequisites to Configuring a Port for SSL	484
■ Configuring SSL Using Configuration Properties Files	485
■ Considerations When Using Configuration Properties	486

Configuration Types for Command Central and Platform Manager OSGI

The OSGI-CCE and OSGI-SPM run-time components support creating configuration instances of the configuration types described in the following table.

Configuration type	Supported on	Use to configure...
COMMON-JAVASYSPROPS	OSGI-CCE, OSGI-SPM	JAVA system properties.
COMMON-JVM-OPTIONS	OSGI-CCE, OSGI-SPM	Extended JVM options.
COMMON-JSW	OSGI-CCE, OSGI-SPM	Java Service Wrapper properties
COMMON-LDAP	OSGI-CCE	A connection to an external LDAP directory.
COMMON-LOCAL-USERS	OSGI-CCE, OSGI-SPM	The internal users for Command Central and Platform Manager.
COMMON-LOG	OSGI-CCE, OSGI-SPM	Logging levels for the logs and log file locations.
COMMON-MEMORY	OSGI-CCE, OSGI-SPM	Common memory settings, such as Initial Heap Size and Maximum Heap Size.
COMMON-PORTS	OSGI-CCE, OSGI-SPM	The HTTP, HTTPS, JMX, JDWP (Debug), and SSH ports. By default, the HTTP, HTTPS, and JMX ports are enabled and the JDWP and SSH ports are disabled. See also " COMMON-PORTS Usage Notes " on page 473.

Configuration type	Supported on	Use to configure...
COMMON-PROXY	OSGI-CCE, OSGI-SPM	The proxy server settings if you must route server requests through a third party server. See also "COMMON-PROXY Usage Notes" on page 473 .
COMMON-SYSPROPS	OSGI-CCE, OSGI-SPM	DEPRECATED. Use COMMON-JAVASYSPPROPS.

COMMON-PORTS Usage Notes

- The following transport protocols allow only one port:
 - JMX
 - SSH
 - JDWP
- The JDWP (Debug) port is disabled by default. You can only update this port using the configuration commands of the CLI, but you cannot add or remove the port. This port is used when the run-time component is started in debug mode using the [sagcc exec lifecycle](#) command.
- Use the JXM port to administer and monitor the JVM KPIs of the Command Central and Platform Manager OSGi components.

COMMON-PROXY Usage Notes

Based on the transport protocol, COMMON-PROXY has the following configuration sub-types:

- COMMON-PROXY-HTTP
- COMMON-PROXY-HTTPS
- COMMON-PROXY-FTP
- COMMON-PROXY-SOCKS
- COMMON-PROXY-ALL - when you want to use the operating system proxy settings instead of the COMMON-PROXY-* configuration.

Important: You cannot edit or delete the COMMON-PROXY-ALL configuration type.

In the CLI configuration commands that you use to modify the proxy server settings, the input XML file that contains the proxy server configuration data must use the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Proxy alias="HTTPS">
  <Enabled>true</Enabled>
  <Protocol>HTTPS</Protocol>
  <Host>hostName</Host>
  <NonProxyHosts>host1,host2</NonProxyHosts>
  <Port>12321</Port>
  <Username>user</Username>
  <Password>secure</Password>
</Proxy>
```

The following table describes the values you must provide for each of the parameters in the input XML file:

Parameter	Description
Alias	The alias name to use for this host:port combination. Specify the same value in the Alias and Protocol parameters.
Enabled	<ul style="list-style-type: none"> ■ true - enabled (default for all proxy configuration types, except COMMON-PROXY-ALL) ■ false - disabled <p>By default, COMMON-PROXY-ALL is disabled. When you enable COMMON-PROXY-ALL, the server uses the proxy settings of the operating system and ignores any COMMON-PROXY-* configurations.</p>
Protocol	The type of protocol to use for the host:port combination. Specify the same value in the Alias and Protocol parameters.
Host	The host name or IP address of the proxy server. For the COMMON-PROXY-ALL configuration type, you can also specify <code>noHost</code>
NonProxyHosts	You can optionally route selected requests directly to their targets, bypassing the proxy. To specify non-proxy hosts, type the fully qualified host and domain name of each server that should receive requests directly. Use <code> </code> as a separator.
Port	The port on which this proxy server listens for requests. Specify a valid port number in the range of [1-65535].
Username	The user name account to access the proxy server.
Password	The password for the specified user name account.

Configuration Types for Command Central and Platform Manager OSGI ENGINE

The OSGI-CCE-ENGINE and OSGI-SPM-ENGINE run-time components support creating configuration instances of the configuration types described in the following table.

Configuration type	Supported on	Use to configure...
COMMON-CREDENTIALS	OSGI-CCE-ENGINE	The user credentials to add and store for a specific alias. You create a configuration instance with the specified alias and then retrieve the stored credentials with the "sagcc get configuration data" on page 218 command to use when connecting to a repository or a remote machine. See also "COMMON-CREDENTIALS Usage Notes" on page 477 .
COMMON-JAAS	OSGI-CCE-ENGINE, OSGI-SPM-ENGINE	The JAAS login modules to use for authentication and authorization, for example to allow authentication against external user stores. See also "COMMON-JAAS Usage Notes" on page 477 .
COMMON-JAAS-REALMS	OSGI-CCE-ENGINE, OSGI-SPM-ENGINE	Configuration settings to use for JAAS realms authentication and authorization, for example to allow authentication against external user stores or Kerberos. See also "COMMON-JAAS-REALMS Usage Notes" on page 478 .

Configuration type	Supported on	Use to configure...
CCE-LAYER-TYPES	OSGI-CCE-ENGINE	Layer definition configuration type.
COMMON-LICENSE	OSGI-CCE-ENGINE	The Command Central license file.
COMMON-LICLOC	OSGI-CCE-ENGINE	Retrieve the location of the Command Central license file.
COMMON-SYSPROPS	OSGI-CCE-ENGINE, OSGI-SPM-ENGINE	The monitoring or inventory parameters, for example the polling interval at which to monitor products for run-time status.
SIN-INTERNAL-GROUPS	OSGI-CCE-ENGINE, OSGI-SPM-ENGINE	The groups in the internal user stores.
SIN-INTERNAL-ROLES	OSGI-CCE-ENGINE, OSGI-SPM-ENGINE	The user roles in the internal user stores.
SPM-NODEID	OSGI-SPM-ENGINE	<p>The internal unique ID of Platform Manager. Command Central manages unique IDs automatically.</p> <p>Before you customize the unique ID of a Platform Manager, run <code>cc list landscape nodes</code> to view the list of IDs already registered with Command Central. Each Platform Manager must have a unique ID in the Command Central landscape.</p>

COMMON-CREDENTIALS Usage Notes

- Following is an example of adding user-defined credentials:

The alias and the credentials details required to create a new instance of the COMMON-CREDENTIALS configuration type is in the `custom_cred.xml` file. To create the new configuration instance for the run-time component with the ID "OSGI-CCE-ENGINE" that is installed in the installation with name "local":

```
sagcc create configuration data local OSGI-CCE-ENGINE COMMON-CREDENTIALS
-i custom_cred.xml
```

- To retrieve the configuration details for the configuration instance with ID "COMMON-CREDENTIALS-myalias" for the run-time component with the ID "OSGI-CCE-ENGINE" that is installed in the installation with name "local", use the following command:

```
sagcc get configuration data local OSGI-CCE-ENGINE COMMON-CREDENTIALS-myalias
```

- The COMMON-CREDENTIALS configuration type has the following default configuration instances that you can retrieve using the [sagcc list configuration instances](#) command, but you cannot delete or edit:
 - COMMON-CREDENTIALS-DEFAULT_ADMINISTRATOR - used when a product or repository requires basic authentication and includes default username and password for the Administrator user.
 - COMMON-CREDENTIALS-NONE - used when no authentication is required, for example to connect to a public github repository.
 - COMMON-CREDENTIALS-TRUSTED - used for trusted authentication with Software AG products or mirror repositories hosted by Platform Manager version 10.0 or higher.

COMMON-JAAS Usage Notes

To modify the JAAS configuration file, use either the Command Central web user interface or [sagcc update configuration data](#). The JAAS configuration files for Command Central and Platform Manager are located here:

- For Command Central

Software AG_directory \profiles\CCE\configuration\security \jaas.config

- For Platform Manager

Software AG_directory \profiles\SPM\configuration\security \jaas.config

You can configure a domain parameter for the InternalLoginModule and the LDAPLoginModule. Command Central uses the value of the domain parameter to determine whether to verify the user against the internal user repository or against an LDAP user store. For example, when you specify the following domain values:

```
com.softwareag.security.jaas.login.internal.InternalLoginModule required
domain="int"
com.softwareag.security.sin.is.ldap.lm.LDAPLoginModule required
domain="sag"
```

If a user logs on with `int\Administrator`, Command Central logs on the user through the Internal login module. If a user logs on with `sag\Administrator`, Command Central logs on the user using the LDAP login module. If you do not configure a domain for the `InternalLoginModule` or the `LDAPLoginModule`, the login module without a domain parameter logs on all users.

COMMON-JAAS-REALMS Usage Notes

To create, update, or delete JAAS realms configuration types, use either the Command Central web user interface, or the CLI "[Options for the Commands](#)" on page .

You must create a separate configuration instance for each JAAS realm, with a unique configuration instance ID in the following format: `COMMON-JAAS-REALMS-realmName`, where *realmName* is the name of the JAAS realm.

The default JAAS realms configuration instance for Command Central and Platform Manager is `COMMON-JAAS-REALMS-Default`.

You can also configure JAAS realms configuration types using a composite template.

The following composite template snippet is an example of how to use the `COMMON-JAAS-REALMS` configuration type to configure JAAS realms for Integration Server:

```
templates:
  is-jaas-config:
    products:
      integrationServer:
        default:
          configuration:
            COMMON-JAAS-REALMS:
              COMMON-JAAS-REALMS-BmKerberos: |
                BmKerberos {
                  com.sun.security.auth.module.Krb5LoginModule required
                  useTicketCache=false
                  doNotPrompt=false
                  debug=true
                  useKeyTab=false;
                };
              COMMON-JAAS-REALMS-BmKerberosKeytab: |
                BmKerberosKeytab {
                  com.sun.security.auth.module.Krb5LoginModule required
                  useTicketCache=false
                  doNotPrompt=false
                  debug=true
                  useKeyTab=true
                  keyTab="config/keytabs/sys21cng.keytab";
                };
```

CCE-LAYER-TYPES Usage Notes

Creates configuration instances of a layer that indicate whether the layer is created from an existing environment.

Use the `sagcc` configuration commands or the Command Central web user interface to create, list, or update configuration instances of the layer definitions.

The default layer definitions are:

- CCE-LAYER-TYPES-INFRA-EXISTING

Defines an infrastructure layer to use when creating a stack from existing environments.

- CCE-LAYER-TYPES-RUNTIME-EXISTING

Defines a run-time layer to use when you want to include existing run-time instances of the products in a stack.

You cannot change the name and type of the default layer definitions.

Run-time Monitoring States for OSGI-CCE and OSGI-SPM

When you run `sagcc get monitoring state`, the OSGI-CCE and OSGI-SPM run-time components return details about the key performance indicators (KPIs) in the following table. Take corrective actions when the KPI threshold approaches critical values.

KPI	Use to monitor...
JVM CPU Load	<p>How much CPU the JVM uses.</p> <p>KPI values:</p> <ul style="list-style-type: none"> ■ Marginal is 80% CPU usage. ■ Critical is 95% CPU usage. ■ Maximum is 100% CPU usage. <p>KPI limitations:</p> <ul style="list-style-type: none"> ■ supported only when running Java 7 or higher ■ not reported when running on HP-UX
JVM Memory	<p>How much JVM memory the run-time component uses.</p> <p>KPI values:</p> <ul style="list-style-type: none"> ■ Marginal is calculates as follows: $\text{MAX}(\text{Maximum} * 80\%, \text{Maximum} - 100)$ A marginal value is when only 20% of the JVM memory is free or less than 100MB of JVM memory is available. ■ Critical is calculated as follows: $\text{MAX}(\text{Maximum} * 95\%, \text{Maximum} - 50)$ A critical value is when only 5% of the JVM memory is free or less than 50MB of JVM memory is available.

KPI	Use to monitor...
	<ul style="list-style-type: none"> Maximum is the amount of memory allocated to the JVM. <p>KPI limitation: the KPI value might be incorrect when running in a 32-bit operating system.</p>
JVM Threads	<p>The number of JVM threads that the run-time component uses.</p> <p>KPI values:</p> <ul style="list-style-type: none"> Marginal and Critical are the current number of the allocated JVM threads plus one thread: current+1. Maximum is the maximum allocated JVM threads plus one thread: MAX+1

Run-time Monitoring States for OSGI-SPM-ENGINE

Note: OSGI-CCE-ENGINE does not support run-time state monitoring.

When you run `sagcc get monitoring state`, the OSGI-SPM-ENGINE run-time component returns details about the key performance indicators (KPIs) in the following table. Take corrective actions when the KPI threshold approaches critical values.

KPI	Use to monitor...
System CPU	<p>The CPU usage of the machine on which Platform Manager is running.</p> <p>KPI values:</p> <ul style="list-style-type: none"> Marginal is 80% CPU usage. Critical is 95% CPU usage. Maximum is 100% CPU usage.
Disk Space	<p>The available disk space on the machine on which Platform Manager is running.</p> <p>KPI values:</p> <ul style="list-style-type: none"> Marginal is 80% of the maximum amount of physical disk space. Critical is 95% of the maximum amount of physical disk space. Maximum is the total amount of physical disk space.

KPI	Use to monitor...
System Memory	<p>The memory usage of the machine on which Platform Manager is running (in MB).</p> <p>KPI values:</p> <ul style="list-style-type: none"> ■ 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.

Server System Properties

This topic contains descriptions of server system properties you can specify for Command Central (CCE) and Platform Manager (SPM) from the Configuration > Java System Properties page in the Command Central web user interface.

You can also set the server system properties for the OSGI-CCE and OSGI-SPM components using the COMMON-JAVASYSPROPS configuration type with the CLI configuration commands or in a composite template.

Command Central uses default values for some of the properties. If a property has a default, it is listed with the description of the property.

Authentication

com.softwareag.platform.management.mirror.trusted.auth.enabled

Indicates which type of authentication to use when connecting to a mirror repository. By default, Command Central uses trusted authentication to connect to a mirror repository. To use basic authentication, set the property to:

```
com.softwareag.platform.management.mirror.trusted.auth.enabled=false
```

Default: true

Migration

com.softwareag.platform.management.client.template.composite.skip.remote.source.archive.check

When migrating from a source archive using a composite template (disconnected migration), the property specifies whether to validate if the source archive exists on the file system of the target Platform Manager node, using SSH. However, if the target Platform Manager node does not have SSH enabled on the remote host, you can skip this validation by setting the property to:

```
com.softwareag.platform.management.client.template.composite.skip.remote.source.archive.check=true
```

Default: `false`

Provisioning

`com.softwareag.platform.management.client.provision.artifacts.job.timeout.minutes`

Configure for OSGI-CCE. The time interval in minutes Command Central waits for a provisioning products job to execute, before cancelling the job.

Default: 180

`com.softwareag.platform.management.client.provision.bootstrap.node.timeout.minutes`

Configure for OSGI-CCE. The time interval in minutes Command Central waits for the bootstrap node jobs to get executed, before cancelling the job.

Default: 120

`com.softwareag.platform.management.client.provision.bootstrap.platform.manager.timeout.minutes`

Configure for OSGI-CCE. The time interval in minutes Command Central waits for a Platform Manager bootstrap job to get executed before cancelling the job.

Default: 120

`com.softwareag.platform.management.client.remote.requests.connect.timeout`

Configure for OSGI-CCE. The time interval in seconds Command Central waits to establish a connection to Platform Manager before terminating the connection attempt.

Default: 900

`com.softwareag.platform.management.client.remote.requests.sync.read.timeout`

Configure for OSGI-CCE. The time interval in seconds in which Command Central synchronizes data with Platform Manager.

Default: 900

`com.softwareag.platform.management.client.template.composite.apply.job.timeout.minutes`

Configure for OSGI-CCE. The time interval in minutes Command Central waits for a composite template apply job to get executed before cancelling the job.

Default: 120

`com.softwareag.platform.management.client.template.composite.node.check.online.poll.timeout.milliseconds`

Configure for OSGI-CCE. The time interval in milliseconds in which to update the status of the Platform Manager nodes.

Default: 2400

com.softwareag.platform.management.client.template.composite.node.check.status.poll.request.enable

When bootstrapping Platform Manager installations using a composite template, the property specifies whether Command Central must check the status of a Platform Manager. This check is additional to the status checks performed by the Command Central monitoring service. You must enable this property only if you experience issues detecting the status of the bootstrapped nodes. By default, the additional check is disabled. To enable the status check, set to:

```
com.softwareag.platform.management.client.template.composite.node.check.status.poll.request.enable=true
```

Default: `false`

com.softwareag.platform.management.template.configuration.type.retry

Specifies which configuration instances of the configuration types to retrieve when applying the template, in addition to the default COMMON-SYSPROPS and COMMON-MEMORY configuration instances. The value is a comma-separated list.

com.softwareag.platform.management.client.template.parallel.exec.nodes

When applying a composite template on installations located on the same physical machine, Command Central applies the inline templates in sequence. This system property enables applying the inline templates in parallel and supports the following values:

- Comma-separated list of node aliases - applies the inline templates in parallel on the listed nodes, for example:

```
com.softwareag.platform.management.client.template.parallel.exec.nodes=nodeAlias1,nodeAlias2
```

In this example, Command Central applies the inline templates on `nodeAlias1` and `nodeAlias2` in parallel.

- ALL - applies the inline templates in parallel on all nodes.

If you do not set the

`com.softwareag.platform.management.client.template.parallel.exec.nodes` property, Command Central applies the inline templates in sequence.

com.softwareag.platform.management.client.template.composite.restart.online.status.poll.timeout.milliseconds

Configure for OSGI-CCE. The time interval in milliseconds in which to update the status of the run-time components.

Default: 2400

com.softwareag.platform.management.client.template.composite.skip.restart.runtimes

Indicates whether to bypass the restart of the run-time components at the end of the composite template application. If the property is set to `false`, Command Central restarts the run-time components at the end of the composite template application. If the property is set to `true`, Command Central does not re-start the run-time components.

Default: `false`

com.softwareag.platform.management.job.thread.pool.size

Configure for OSGI-CCE. The number of jobs that can run in parallel. The number of jobs corresponds to the number of nodes that Command Central processes in parallel. For example, if the thread pool size is set to 10 and you have 100 nodes in the list, the nodes are processed 10 at a time in parallel, in the order in which they are listed.

Default: 40

com.softwareag.platform.management.job.timeout

Configure for OSGI-CCE and OSGI-SPM. The time interval in seconds Command Central waits for a generic job to get executed before cancelling the job.

Default: 4800

com.softwareag.platform.management.mirror.job.timeout

Configure for OSGI-CCE and OSGI-SPM. The time interval in minutes Command Central waits for a mirror repository job to get executed, before cancelling the job.

Default: 720

com.softwareag.plm.sum.cc.override.resolve.validation

Configure for OSGI-SPM. Specifies whether to validate unresolved fix dependencies when applying a product fix. By default, Command Central validates unresolved fix dependencies. To disable this validation:

```
com.softwareag.plm.sum.cc.override.resolve.validation=false
```

Default: `true`

Prerequisites to Configuring a Port for SSL

In a production environment, the HTTP port does not provide protection when sending sensitive information, for example administrator passwords, over the network. Software AG recommends disabling the HTTP port in a production environment and using an HTTPS port to ensure secure communication with the Command Central and Platform Manager servers.

Command Central provides a default keystore and truststore that are available after installing Command Central. The keystore and truststore are files that function as repositories for storage of keys and certificates necessary for Secure Socket Layer (SSL) authentication, encryption/decryption, and digital signing/verification services. The default keystore and truststore contain signed certification authority (CA) certificates that the Command Central server uses to validate client certificates.

You can replace the Command Central default keystore and truststore files with custom files. For information about creating keystores and truststores, importing keys and

certificates into keystores and truststores, and other operations with these files, refer to the documentation for your certificate management tool.

Before configuring an HTTPS port, you must configure the Command Central server to use SSL, using the Command Line tool.

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 170.

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 "[Common Options](#)" on page 167.

5. Save the custom cc.properties file.

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 `sagcc` 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

29 Using the Command Central and Platform Manager Logs

■ Command Central Logs	490
■ Platform Manager Logs	491
■ Using the Correlation ID	493
■ Logging Levels	494
■ Changing the Log Configuration Settings	494
■ Deleting Logs	495

Command Central Logs

The Command Central logs contain information about operations and errors that occur on the Command Central server, such as starting run-time components managed by Command Central and applying a composite template. The Command Central default and wrapper logs rotate by size, based on the size limit configured for the log file. The default size limit is 10MB. When Command Central rotates the log files, Command Central appends a number to the log file name, but the current log files are always named `default.log` and `wrapper.log`.

Command Central uses this format for log entries:

time_stamp log_level correlation_id message_text

For a description and details about the correlation ID, see ["Using the Correlation ID" on page 493](#).

The following sections describe the Command Central logs available from the Logs tab in the Command Central web user interface.

Default Log

Use the default log to monitor the progress of Command Central operations and check about warning or error messages that might signal impending or actual failure.

Following is an example snippet from a Command Central default log:

```
2016/11/15 14:12:20 INFO #4 Layer: installation Node: mirror_10.0
  Templates: [product-template, fix-template]
2016/11/15 14:12:20 INFO #4 stripe template [product-template]
  node [mirror_10.0] layer [installation] begin
2016/11/15 14:12:20 INFO #4 pre actions template [product-template]
  node [mirror_10.0] layer [installation] begin executing...
2016/11/15 14:12:20 INFO #4 pre actions template [product-template]
  node [mirror_10.0] layer [installation] finished executing
2016/11/15 14:12:20 INFO #4 import template [product-template]
  node [mirror_10.0] layer [installation] begin
2016/11/15 14:12:20 INFO #4 import template [product-template]
  node [mirror_10.0] layer [installation] success
2016/11/15 14:12:20 INFO #4 import template [product-template]
  node [mirror_10.0] layer [installation] end
2016/11/15 14:12:20 INFO #4 apply template [product-template]
  node [mirror_10.0] layer [installation] initiated with options [PRODUCTS]
2016/11/15 14:12:20 INFO #4 apply template [product-template]
  node [mirror_10.0] layer [installation] scheduled
2016/11/15 14:15:59 ERROR # [CCEMONE0021] Error polling SPM node
[mirror_10.0]. Marking all states on the node as unknown.
2016/11/15 14:16:37 INFO #4 Waiting for mirror_10.0 to become ONLINE...
```

Wrapper Log

Use the wrapper log to troubleshoot and debug issues that occur during Command Central operations. This log includes DEBUG messages with Command Central code-level statements and WARN and ERROR messages from the code of other facilities.

Rest API Log

Use this log to troubleshoot all requests and responses from the Command Central REST API client and server.

Command Central uses this format for the request/response entries in this log:

number > *request*

number < *response*

Each request and its response have the same number before the greater/less than signs, for example:

```

364 > GET https://rubicon03:8091/cce/monitoring/alerts/?nodeAlias=local&
runtimeComponentId=OSGI-CCE&includeChildren=true
364 > accept: application/vnd.sagcc.job+json,application/json
364 > accept-encoding: gzip, deflate
364 > accept-language: en-US,en;q=0.5
364 > connection: keep-alive
364 > content-type: application/json
364 > csrfpreventiontoken: un1d8qeuk3el4iba3gg1b4tf6
364 > donottrynextauth: true
364 > host: rubicon03:8091
364 > referer: https://rubicon03:8091/cce/web/
364 > user-agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0)
Gecko/20100101 Firefox/45.0
2016/11/15 15:46:52 INFO CCAApplication 364 * Server responded with
a response on thread http-bio-8091-exec-7
364 < 200
364 < Content-Type: application/json

```

Bootstrap Log

Use this log to monitor or troubleshoot operations when bootstrapping Platform Manager. The Platform Manager bootstrap installer creates a separate bootstrap log for each executed bootstrap operation. The name of each bootstrap log file uses the format `bootstrap.n.log`, where *n* is an incremental number that indicates a separate bootstrap operation.

Platform Log

This log contains messages logged from the OSGi framework.

Platform Manager Logs

The Platform Manager logs contain information about operations and errors that occur on Platform Manager, such as updating configuration settings for a product instance and processing inline templates. The Platform Manager default and wrapper logs rotate by size, based on the size limit configured for the log file. The default size limit is 10MB. When Platform Manager rotates the log files, Platform Manager appends a number to the log file name, but the current log files are always named `default.log` and `wrapper.log`.

Platform Manager uses this format for log entries:

```
time_stamp log_level correlation_id message_text
```

For a description and details about the correlation ID, see ["Using the Correlation ID" on page 493](#).

The following sections describe the Platform Manager logs available from the Logs tab in the Command Central web user interface.

Default Log

Use the default log to monitor the progress of Platform Manager operations and check about warning or error messages that might signal impending or actual failure.

Following is an example snippet from a Platform Manager default log:

```
2016/11/15 13:43:25 INFO #11 There are "4" Prerequisites to be applied.
2016/11/15 13:43:25 INFO #11 There are "1" configurations to be applied.
2016/11/15 13:43:25 INFO #11 Applying prerequisite "Empty prerequisite"
  to each configuration instance...
2016/11/15 13:43:25 INFO #11 Processing Configuration instance
  "ConfigurationInstanceDTO [id=COMMON-SYSPROPS, displayName=null,
  description=null, configurationTypeId=COMMON-SYSPROPS]"
2016/11/15 13:43:25 INFO #11 Applying prerequisite "Empty prerequisite"
  for Runtime Component "OSGI-CCE" of configuration instance "COMMON-SYSPROPS".
2016/11/15 13:43:25 INFO #11 Prerequisite - "Empty prerequisite" - applied
2016/11/15 13:43:25 INFO #11 The configuration "COMMON-SYSPROPS" will be updated
2016/11/15 13:43:25 INFO #11 Updating an existing configuration with id:
  COMMON-SYSPROPS for component with id: OSGI-CCE
2016/11/15 13:43:25 INFO #11 Configuration instance "COMMON-SYSPROPS" is applied
  for runtime component "OSGI-CCE"
2016/11/15 13:43:25 INFO #11 Invalidating Inventory and Configuration caches...
2016/11/15 13:43:25 INFO #11 There are "0" configurations to be applied.
```

Wrapper Log

Use the wrapper log to troubleshoot and debug issues that occur during Platform Manager operations, for example when processing inline templates. This log also includes WARN and ERROR messages logged by the instance managers of managed products.

Rest API Log

Use this log to troubleshoot all requests and responses from the Platform Manager REST API client and server.

Platform Manager uses the same format for entries in this log as the format of the Command Central REST API log. For details about the format, see ["Command Central Logs" on page 490](#).

SD Provisioning Log

This log contains messages logged when installing products through Command Central and Platform Manager.

Provision Script Install Log

This log contains messages logged from the Software AG Installer API for products that get installed from Installer image files.

SUM Provisioning Log

This log contains messages logged when installing fixes through Command Central and Platform Manager. The messages in this log are generated by the Update Manager API, but stored in this Platform Manager log.

SUM Debug Logs

The following Update Manager debug log is also available from the Command Central web user interface:

`sum_debug_timestamp_n.log` - where *n* is an incremental number. This log is generated by the Update Manager to monitor or troubleshoot the Update Manager operations.

Using the Correlation ID

The correlation ID is a number that is automatically generated for every request that the Command Central server processes. Command Central uses the correlation ID to track each request.

The correlation ID is displayed in the Command Central and Platform Manager logs in the following format:

```
#number
```

The number is an integer that is incremented for each new request included in the log. The numbering in the logs is reset to one after restarting Command Central.

For example:

```
2016/11/15 14:15:35 INFO #4 End of Template Operation APPLY
```

The correlation ID is not an error code, but it helps you establish which conditions and steps have lead to an issue. You can use the number of the correlation ID as a search filter when tracing an operation in the Command Central and Platform Manager logs. For example, you can use the correlation ID to track a specific operation during the processing of a composite template. The operation will have the same correlation ID in the Command Central and Platform Manager logs.

Requests between Command Central and Platform Manager

The correlation ID is not generated for non-HTTP requests processed by Command Central. For example, the scheduled polling requests from Platform Manager to an Integration Server instance for monitoring data are not HTTP requests and do not have correlation IDs. The log entries for such requests include the # sign without a number. For example:

```
2016/11/15 14:15:37 INFO # Monitoring is ENABLED and will start to poll
```

```
after 30 seconds every 30 seconds.
```

In the Platform Manager logs, requests from the Command Central server to Platform Manager that poll for changes at scheduled intervals are logged with the constant correlation ID of #00000. For example:

```
2016/11/15 14:08:30 INFO #00000 SUM Update Manager begin initialize()
```

Logging Levels

The logging levels that you can specify for the Command Central and Platform Manager logs are listed below. Each logging level includes the indicated type of message plus all messages from the levels above it (for example, the Info level includes Error, Warn, and Info messages).

- ERROR
- WARN
- INFO
- DEBUG
- TRACE

The default logging level for the default logs is INFO and for the wrapper logs the default is DEBUG.

You can change the logging level when you want to increase or decrease the amount of information to include in the logs. For example, you can temporarily increase the level of detail written to a log from DEBUG to TRACE to find the cause of an error or performance problem, and return to DEBUG level after resolving the problem. For information about changing the log configuration settings, see ["Changing the Log Configuration Settings" on page 494](#).

Changing the Log Configuration Settings

You can change the default configuration settings for a log, for example when you want to specify a different location for a log file. You use the Command Central CLI configuration commands to change the configuration settings for the Command Central and Platform Manager logs, except for the wrapper logs.

1. Use the following Command Central CLI command to get the log configuration data for a node:

- For the Command Central logs:

```
sagcc get configuration data node_alias OSGI-CCE COMMON-LOG-log_config.xml  
-o log.xml
```

- For the Platform Manager logs:

```
sagcc get configuration data node_alias OSGI-SPM COMMON-LOG-log_config.xml  
-o log.xml
```

The log.xml file from the output of the command contains the log configuration settings.

2. Open the log.xml file in a text editor and change the values of the parameters as required.
3. In the Command Central CLI, update the log configuration settings using this command:

- For the Command Central logs:

```
sagcc update configuration data node_alias OSGI-CCE COMMON-LOG-log_config.xml
-i log.xml
```

- For the Platform Manager logs:

```
sagcc update configuration data node_alias OSGI-SPM COMMON-LOG-log_config.xml
-i log.xml
```

Changing the Wrapper Logs Configuration

To change the configuration settings for the Command Central and Platform Manager wrapper logs, you must edit the custom_wrapper.config file located in:

- For Command Central: Software AG_directory/profiles/CCE/configuration
- For Platform Manager: Software AG_directory/profiles/SPM/configuration

For information about the logging configuration properties in the Java Service Wrapper configuration file, see Software AG Infrastructure Administrator's Guide and <https://wrapper.tanukisoftware.com>

Deleting Logs

Command Central rotates log files automatically based on size, not time. The actions you do to delete the logs depend on the type of log file rotation.

Size-based Log Rotation (Default)

To delete logs when log rotation is size-based, follow the steps in the topics on this page.

Delete Command Central Logs

With the size-based log rotation, you must locate the log files on the file system and delete them as follows:

1. To stop the CCE instance:
 - In the Command Central web user interface, click **CCE**. On the Overview tab, click **Lifecycle Actions**  and select **Stop**.
 - In the CLI, run the "[sagcc exec lifecycle](#)" on [page 356](#) command.

2. Locate the log files you want to delete.
 - In *Software AG_directory*\profiles\CCE\logs

Tip: In the Command Central web user interface, click **CCE** and go to the Logs tab. Moving the mouse pointer over a log alias shows the location of the log on the file system.

- After migrating an environment, log files are also located in *Software AG_directory*\install\logs
- Check the logging configuration in *Software AG_directory*\profiles\CCE\configuration\logging\log_config.xml

Tip: In the Command Central web user interface, click **CCE** and go to **Configuration > Java Service Wrapper**.

If the logging configuration includes custom log appenders, check the locations of the log appender files.

3. Delete the log files.

Delete Platform Manager Logs

With the size-based log rotation, you must locate the log files on the file system and delete them as follows:

1. To stop the SPM instance, from the command prompt:
 - a. Change directory to *Software AG_directory* \profiles\bin
 - b. Run shutdown.bat|sh
2. Locate the log files you want to delete.
 - In *Software AG_directory*\profiles\SPM\logs

Tip: In the Command Central web user interface, click **SPM** and go to the Logs tab. Moving the mouse pointer over a log alias shows the location of the log on the file system.

- After migrating an environment, log files are also located in *Software AG_directory*\install\logs
- Check the logging configuration in *Software AG_directory*\profiles\SPM\configuration\logging\log_config.xml

Tip: In the Command Central web user interface, click **SPM** and go to **Configuration > Java Service Wrapper**.

If the logging configuration includes custom log appenders, check the locations of the log appender files.

3. Delete the log files.

Time-based Log Rotation

Important: If you use time-based rotation, the Platform Manager logs will not rotate when Platform Manager is stopped and the Command Central logs will not rotate when Command Central is stopped.

You can change the logging configuration to rotate the Command Central and Platform Manager logs at specific time intervals, such as monthly, weekly, or daily.

To change the log rotation to time intervals, go to the log4j documentation and follow the instructions for [DailyRollingFileAppender](#) or [rolling.RollingFileAppender](#)

For information about how to update the logging configuration, see "[Changing the Log Configuration Settings](#)" on page 494.

30 Introduction to Command Central REST API

■ About Command Central REST API	500
■ Securing Command Central REST API	500
■ Command Central REST API Resources	500
■ Supported Media Types	501
■ HTTP Response Codes	502
■ Summary of REST Services	502

About Command Central REST API

Command Central REST API is a web services API that supports all functions provided by Command Central. Command Central REST API is for developers who want to build functionality on top of Command Central. Along with the Command Central web user interface and command line tool, Command Central REST API enables you to use Command Central to configure, manage, and administer one or more installations of the Software AG product suite in your enterprise.

Securing Command Central REST API

At present Command Central REST API supports basic and trusted authentication. Command Central REST API uses the same credentials to authenticate a user as the ones you provide for the Command Central web user interface and command line tool. For more information about setting up security credentials, see *Software AG Command Central Help*.

Session Management

Command Central REST API is stateless. However, the REST API supports HTTP sessions for stateful clients, for example browsers. When the client does not support session management, for example does not support JSESSION cookie, the client *must* submit `DoNotCreateSession: true` HTTP request header to prevent creating a new session for each client request.

Command Central REST API Resources

The Command Central REST API resources use the HTTP methods GET, POST, PUT, and DELETE to execute different operations. Each Command Central REST API resource is identified by a named URI that has the following base endpoint:

```
http|https://ccehost:cceport/cce/service_name/[resource[/subresource/]...]
```

where

ccehost is the name of the host machine where you have installed Command Central.

cceport is the port number where the Command Central instance is running.

service_name is the name of the Command Central REST API service.

[*resource[/subresource/]...*] (optional) is different for each Command Central REST API service.

Command Central REST API uses the [Jersey RESTful framework](#) that supports the Web Application Description Language (WADL). The WADL page for each Command

Central REST API service contains details about the resources, such as resource URI, supported media types, XML schemas for data structure, and HTTP methods. For more information about the Jersey RESTful framework and WADL, see the Jersey framework documentation.

For more information about the Command Central REST API resources, see:

- The WADL page for the Command Central REST API services. You can find the name for each REST service and the URI for the WADL page in "[Summary of REST Services](#)" on page 502.
- In the Command Central command line tool, execute a command with the `--debug` option. This option returns REST API request and response details, such as URI, HTTP method, content type, content body, and HTTP response code. For information about the Command Central commands, see *Software AG Command Central Help*.

Testing Command Central REST API Resources

Use one of the following methods to test a Command Central REST API resource:

- In the Command Central command line tool, execute a command with the `--debug` option (recommended). For example:

```
sagcc list landscape nodes --debug -f json -u Administrator
-p manage -s http://localhost:8090/cce
```

- Use REST API clients browser plug-ins for Firefox and Chrome.
- Use a third-party command line tool, for example [cURL](#)

Example commands using cURL:

```
curl -u Administrator:manage -X GET -H "Accept: application/json" http://localhost:8090/cce/1
curl -u Administrator:manage -X GET -H "Accept: application/xml" http://localhost:8090/cce/1a
```

Supported Media Types

Command Central REST API resources support at least one of the following media types:

- application/xml
- application/json
- text/plain

The following media types are supported for all GET resources that return lists of objects:

- text/csv
- text/tab-separated format

For information about the media types supported by each resource and method, see the REST service WADL pages. You can find the URI for each REST service in ["Summary of REST Services" on page 502](#).

HTTP Response Codes

The Command Central REST API response returns an HTTP response code that indicates success or error of the requested operation.

HTTP response code	Description
2xx	Success.
4xx	Client error. Correct the request data and retry.
5xx	Server error.

The body of the 4xx and 5xx responses normally contains additional information about the error, such as error code, description, and action. Some error messages include a nested error cause.

For information about the HTTP response codes supported by each resource and method, see the REST service WADL pages. You can find the URI for each REST service in ["Summary of REST Services" on page 502](#).

Summary of REST Services

The following table lists the Command Central REST API services that you can locate at `http(s)://<ccehost>:<cceport>/cce/application.wadl`.

Service Name	Description
Administration	Executes custom administration actions for a product or run-time component.
Configuration	Manages configuration for run-time components.
Diagnostic	Retrieves information from the log files that a run-time component supports.
Inventory	Retrieves information about products, run-time components, and fixes.

Service Name	Description
Job Manager	Lists information about long-running jobs.
Landscape	Manages environments and installations.
License Tools	Creates and manages license reports to verify product license compliance.
Lifecycle	Executes an action to start, stop, pause, and/or resume run-time components.
Monitoring	Reports run-time component status, state, and alerts.
Provisioning	Bootstraps Platform Manager locally and remotely. Installs and uninstalls products and fixes. PREVIEW FEATURE. Installs and uninstalls assets.
Repository	Manages product, fix, and assets repositories.
Security	Manages security credentials.
Stacks	PREVIEW FEATURE. Creates stacks and layers.
Template	Manages templates of environments and installations.
