

# Software AG Command Central Help

Version 10.11

October 2021

---

This document applies to Software AG Command Central 10.11 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 © 2013-2023 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 <https://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 <https://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 <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

**Document ID: CC-OH-1011-20231110**

# Table of Contents

<b>About this Guide</b> .....	7
Document Conventions.....	8
Online Information and Support.....	8
<b>1 Getting Started with Command Central</b> .....	<b>11</b>
What is Software AG Command Central?.....	12
Install Command Central.....	13
Start or Stop Command Central.....	15
Configure HTTP/S Proxy.....	16
Create Credentials Aliases.....	16
Connect to Software AG Repositories.....	17
Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes.....	17
Import Product License Keys for Instances or Components.....	19
Register Bootstrap Installers in Command Central.....	20
Additional Configuration for Command Central in a Production Environment.....	21
Upgrade Command Central.....	24
<b>2 Creating, Managing, or Upgrading Standalone Product Installations</b> .....	<b>27</b>
Understanding Standalone Product Installations.....	29
Understanding Product and Fix Repositories.....	30
Understanding Asset Repositories.....	31
Default and Custom Product Administrator User Passwords.....	31
Connect an Existing Product Installation to Command Central and Add It to Environments.....	32
Create a New Standalone Installation.....	33
Create and Update Product Instances.....	41
Change the Status of an Instance or Component.....	42
Install Fixes.....	42
Install Support Patches.....	43
Compare Products, Fixes, and Instances.....	44
View KPIs and Alerts for an Instance or Component.....	44
Add an Environment to a Landscape.....	44
Configure Shared Secrets to Encrypt Instance and Component Passwords.....	45
Encrypt User Passwords for Product Configurations.....	45
Group Product Instances or Components.....	46
Monitor the Licensing State for a Landscape.....	46
Upgrade a Standalone Product Installation.....	46
Understanding License Metering (DEPRECATED).....	48
<b>3 Creating or Upgrading Software Stacks</b> .....	<b>49</b>
Understanding Software Stacks.....	50
Understanding Product and Fix Repositories.....	51
Create a Software Stack.....	52

---

Create Software Stacks from Existing Environments.....	57
Rename a Software Stack or Layer.....	58
Upgrade Software Stacks.....	58
<b>4 Common Use Cases with Command Central.....</b>	<b>59</b>
How to Get Templates for Upgrade.....	60
Install or Upgrade Product Installations with Composite Templates.....	63
Install or Upgrade Product Installations with Stacks.....	67
Update Licenses on a Number of Nodes with Templates.....	69
<b>5 Command Central GitHub Projects.....</b>	<b>71</b>
Automate Command Central Installation, Configuration, and Upgrade.....	72
Default Templates on GitHub.....	72
<b>6 Using Micro Templates.....</b>	<b>73</b>
What is a Micro Template?.....	74
When to Use Micro Templates and When a Composite Template?.....	74
Connecting to the Default Templates Repository.....	75
Designing Micro Templates.....	75
Using the Sample Micro Templates.....	76
Creating a User-Defined Micro Template.....	76
Exporting the Platform Manager of an Installation to a Template.....	77
Exporting an Installed Product Instance to a Micro Template.....	79
Adding Configurations in an Exported Micro Template.....	81
Exporting the Database for Products in an Installation to a Template.....	81
Validating a Micro Template.....	82
<b>7 Using Composite Templates.....</b>	<b>83</b>
About Automated Provisioning of Environments.....	84
What is a Composite Template?.....	84
Understanding the Composite Template Definition.....	85
Creating a Custom Composite Template.....	116
How Command Central Processes a Composite Template for Installations of Release 10.3 or Higher.....	116
How Command Central Processes a Composite Template for Installations of Release 10.1 or Lower.....	119
Importing a Template into Command Central.....	120
Creating a New Environment Using a Composite Template.....	120
Updating Command Central to the Latest Fix.....	121
Updating a Provisioned Environment Using the Same Composite Template.....	122
Migrating to a Different Host with Cloned Database.....	122
Migrating Using Disconnected Migration.....	124
Migrating a Product Instance.....	125
Migrating Platform Manager Using a Template.....	127
Monitoring the Status of a Composite Template.....	129
Validating a Composite Template.....	129
Correcting a Failed Composite Template Apply Operation.....	130
Composite Templates for 9.x Installations.....	131

---

<b>8 Bootstrapping Platform Manager on a Remote Machine Using a Template.....</b>	<b>145</b>
Bootstrapping Platform Manager Using the Default SSH and WinRM Templates.....	146
Bootstrapping on a Remote Machine with a Substitute User.....	146
<b>9 Encrypting Passwords in Templates.....</b>	<b>149</b>
Encrypting Passwords in Templates.....	150
<b>10 Managing Database Components Using a Template.....</b>	<b>151</b>
Managing Database Components Using a Template.....	152
Supported Database Component Configurator Parameters.....	156
<b>11 Deploying Command Central Assets to Integration Cloud.....</b>	<b>159</b>
About Deploying Command Central Assets to webMethods Cloud Container (Deprecated).....	160
Defining Environment Variables for Platform Manager Assets.....	160
How Platform Manager Deploys Assets with Dependencies.....	161
<b>12 Command Central Composite Assets.....</b>	<b>163</b>
About Command Central Composite Assets.....	164
Adding Assets to the Build Source Directory.....	164
Setting Build Properties for Command Central Assets.....	164
Building Command Central Composite Assets.....	166
<b>13 Using the Command Line Interface.....</b>	<b>167</b>
Installing CLI as a Remote Client.....	169
Displaying Help for the Command Line Interface.....	169
Displaying the Version of the Command Line Interface.....	170
Executing Command Central Commands.....	170
Executing Platform Manager Commands.....	171
Getting Familiar with Using Commands.....	171
Return Codes from Command Execution.....	173
Invoking Commands from Scripts.....	173
Managing Database Components Using CLI Commands.....	188
Options for the Commands.....	193
Administration Commands.....	222
License Inventory Commands.....	226
Configuration Commands.....	229
Diagnostics Logs Commands.....	260
Instance Management Commands.....	266
Inventory Commands.....	273
Jobmanager Jobs Commands.....	301
Landscape Commands.....	306
License Keys Commands.....	325
License Reports Commands.....	334
Lifecycle Commands.....	348
Monitoring Commands.....	352
Provisioning Bootstrap Installers Commands.....	359

Provisioning Fixes Commands.....	364
Provisioning Products Commands.....	368
Repository Commands.....	372
Resources Commands.....	413
Security Credentials Commands.....	414
Stacks and Layers Commands.....	425
Template Commands.....	450
<b>14 Configuring Command Central and Platform Manager.....</b>	<b>483</b>
Configuration Types for Command Central and Platform Manager OSGI.....	484
Configuration Types for Command Central and Platform Manager OSGI ENGINE.....	487
Run-time Monitoring States for OSGI-CCE and OSGI-SPM.....	491
Run-time Monitoring States for OSGI-SPM-ENGINE.....	492
Server System Properties.....	493
Using the Secure Socket Layer (SSL) Protocol and Certificates in Command Central.....	499
Preparing to Replace the Default Keystore and Truststore.....	499
Update the SSL Connection Settings for the CLI.....	502
Considerations When Using Configuration Properties.....	503
<b>15 Introduction to Command Central REST API.....</b>	<b>505</b>
About Command Central REST API.....	506
Securing Command Central REST API.....	506
Command Central REST API Resources.....	506
Supported Media Types.....	507
HTTP Response Codes.....	507
Summary of REST Services.....	508
<b>16 Logging and Troubleshooting.....</b>	<b>511</b>
Using the Command Central and Platform Manager Logs.....	512
Activity Appears to Stop During File Download.....	520
Bootstrapper Fails with Package Error Such as "installer.jar not found".....	520
Expected Product-specific Features are Not Available.....	520
Cannot Connect to Repositories Due to Invalid Credentials.....	520
Cannot Create Mirror Repository Because of Network Issues.....	520
Command Central Cannot Connect to Platform Managers.....	521
Values with Special Characters in a Template YAML File.....	521
Collecting Diagnostic Information with the Syscap UNIX Shell Script.....	521
Applying a Template Fails Because Product Configurations Fail to Apply.....	522
Setting the Values of Java System Properties in the Command Central Web User Interface and CLI.....	522
Remote Operations Run by Command Central over SSH get Terminated.....	522
Command Central Cache Takes too Much Disk Space.....	522

# About this Guide

- Document Conventions ..... 8
- Online Information and Support ..... 8

---

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

## Document Conventions

---

Convention	Description
<b>Bold</b>	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies:  Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies:  Text you must type in. Messages displayed by the system. Program code.
{ }	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 and Support

---

### Software AG Documentation Website

You can find documentation on the Software AG Documentation website at <https://documentation.softwareag.com>.

### Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to [empower@softwareag.com](mailto:empower@softwareag.com) with your name, company, and company email address and request an account.

---

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

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](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at [https://empower.softwareag.com/public\\_directory.aspx](https://empower.softwareag.com/public_directory.aspx) and give us a call.

## **Software AG Tech Community**

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

- Access product documentation, if you have Tech Community 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.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/u/softwareag> and discover additional Software AG resources.



# 1 Getting Started with Command Central

---

■ What is Software AG Command Central? .....	12
■ Install Command Central .....	13
■ Start or Stop Command Central .....	15
■ Configure HTTP/S Proxy .....	16
■ Create Credentials Aliases .....	16
■ Connect to Software AG Repositories .....	17
■ Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes .....	17
■ Import Product License Keys for Instances or Components .....	19
■ Register Bootstrap Installers in Command Central .....	20
■ Additional Configuration for Command Central in a Production Environment .....	21
■ Upgrade Command Central .....	24

## 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. You can also automate tasks by using commands to remotely execute actions from a terminal or custom script. Command Central supports continuous integration servers such as Jenkins, and generic configuration management tools such as Puppet and 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) and install fixes (9.7 and later).

The upgrade procedure for software stacks is simpler than the procedure for standalone product installations. For stacks, Command Central automatically generates composite templates that you can then use to upgrade a single environment or to automate the upgrade of multiple environments. If you are not yet using stacks, you can create stacks from the installations you want to upgrade. For instructions on all of these topics, see *Upgrading Software AG Products On Premises*.

When the host servers in your environments do not have internet access, you can install one Command Central on a machine that has internet access. Then you install one or more Command Centrals to manage your development and test environments, and one or more Command Centrals to manage the production environments. The Command Central with internet access can connect to the Software AG repositories on Empower so you can download the products and fixes you have licensed. The hosts in each environment then get the Software AG products and fixes through the mirror repositories registered in the Command Central that manages that environment. The Command Centrals that manage your environments must have access:

- To the Command Central that connects to Empower
- 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.

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.

---

## Install Command Central

---

You install Command Central, Platform Manager, and the Command Central command line interface (CLI) using the Command Central bootstrapper. Software AG refreshes the bootstrapper every time new fixes are released for those components, so always use the latest bootstrapper to install Command Central, Platform Manager, and the Command Central CLI.

At any time after you have installed Command Central, Platform Manager, and the Command Central CLI, you can install new fixes that are released for those components by downloading the latest bootstrapper for the same Command Central release and running it against the existing Command Central installation.

Installation requirements are as follows:

- The machine on which you install Command Central must have the following:
    - At least 5GB of disk space, 2GB of RAM, and 4 CPU cores.
    - An additional 50G if you are going to use Command Central to install your products, install fixes on your products, or upgrade your products.
    - Access to product host machines.
  - The operating system on the Command Central host machine must be among those listed in *System Requirements for Software AG Products*. Windows operating systems must have the latest Windows updates. In particular, installation on Windows 8.1 and Windows Server 2012 R2 will fail if Microsoft update KB2919355 from April 2014 is not present.
  - Command Central must be the only product in its installation directory, so you can easily upgrade Command Central even when you are not upgrading your products.
1. Review the Software AG product license agreement at [http://documentation.softwareag.com/legal/general\\_license.txt](http://documentation.softwareag.com/legal/general_license.txt). You will have to accept the agreement when you run the bootstrapper.
  2. Follow the instructions in your installation email from Software AG to download the Command Central bootstrapper for your operating system.

Beginning with release 10.7, the .exe file of the bootstrapper is deprecated. Download the .bat file to install Command Central on a Windows operating system.

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

3. On Windows, create an installation user account with Windows system administrator privileges. On UNIX, create a non-root installation user account with full read and write permissions to the target installation directory. The account will own all files you install.
4. Log on to the target machine under the account you just created.
5. Launch a command shell. On Windows, launch using Run as Administrator.

- Run the command below using the privileges or permissions stated earlier. The bootstrapper command provided below uses default values, where `--accept-license` means you accept the Software AG product license agreement (available for review at [https://documentation.softwareag.com/legal/general\\_license.txt](https://documentation.softwareag.com/legal/general_license.txt)). If you want to override the default values, see “Override Bootstrapper Command Default Values” on page 14.

**Note:**

On Windows, if you do not run the command under a user account with system administrator privileges, the console window will close automatically and you will not be able to see output describing success or errors, nor will you be able to see `-help` output.

The installation process produces many messages. To capture all messages, send the output to a file by specifying `> bootstrapper.out` on the command below (or, if you prefer, increase your buffer before running the command).

```
cc-def-release-fixnumber-operating_system.{bat|sh} --accept-license  
[> bootstrapper.out]
```

- Write down the Command Central URL and authentication credentials that are listed in the command shell so you can log on.

## 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`:

```
cc-def-10.3-fix1-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-fix3-lnxamd64.sh  
cc-def-10.2-fix3-lnxamd64.sh -d /opt/sagcc -H cchost.com -c 9090 -C 9091  
-s 9092 -S 9093 -p $SuperCCAdmin --accept-license
```

## Override Bootstrapper Command Default Values

The following table shows arguments you can specify on the Command Central bootstrapper command to override the default values.

**Note:**

Make sure to run the bootstrapper `.bat` or `.sh` script using the privileges or permissions stated earlier. On Windows, if you do not run the `.bat` under a user account with system administrator privileges, the console window will close automatically and you will not be able to see output describing success or errors, nor will you be able to see `-help` output.

Argument	Value	Default
-H <i>host_name</i>	DNS name or IP address for the machine on which to install Command Central.	Machine on which you are running the bootstrapper.
-d <i>path</i>	Full path to the installation directory in which to install Command Central.	Directory from which you are running the bootstrapper.
-c <i>port_number</i>	HTTP port to use for Command Central.	8090
-C <i>port_number</i>	HTTPS port to use for Command Central.	8091
-s <i>port_number</i>	HTTP port for Command Central to use to communicate with the local Platform Manager.	8092
-S <i>port_number</i>	HTTPS port for Command Central to use to communicate with the local Platform Manager.	8093
-p <i>password</i>	Password to use for the Command Central Administrator user account. In a production environment, Software AG strongly recommends replacing the default with a strong, custom password.	manage
	<p><b>Note:</b> If you are installing on Windows, and want to use special characters like ampersands (&amp;), specify <code>--base64-pass</code> instead of <code>-p</code>. This argument allows you to pass a base64-encoded password.</p>	
-P <i>sudo_password</i>	On UNIX, sudo password to use to register daemons for Command Central and Platform Manager.	
	<p><b>Note:</b> You can register daemons at a later time by re-running the bootstrapper with this argument.</p>	

## 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.

## Configure HTTP/S Proxy

---

To connect to a Software AG repository, go to the Command Central that has Internet access and define a connection from the local Platform Manager to the repository. If your internet connection is routed through a proxy server, the connection must go through a proxy server. Go to **Environments > All > Instances > SPM > Configuration > Proxy** to configure the proxy.

## Create Credentials Aliases

---

To connect to product, fix, or asset repositories from Command Central, you must supply the appropriate credentials.

1. Go to Empower and log on with the user name and password from the installation email sent to you by Software AG. Then go to **Products & Documentation > Download Products > Software Downloads > Software Download Center**. This will display the Software AG product license agreement for first time users. If you accept, you will be able to connect to the Software AG Software Download Center.
2. Go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials**, click .
3. The following table describes the steps to follow to create aliases for credentials to use to connect to repositories.

Repository Type	Steps
Software AG product and fix repositories	<p>Click <b>Username and password</b> and create an alias for your Empower user name and password.</p> <p>If you want additional users to be able to connect to Software AG product and fix repositories, follow these steps:</p> <ol style="list-style-type: none"><li>a. Go to <b>Products &amp; Documentation &gt; Download Products &gt; Software Downloads</b>.</li><li>b. Click the link for the region or regions to which the additional users will need to connect.</li><li>c. In the email that displays, list the users by their company email addresses. Software AG will send each of those users an email containing Empower credentials with the necessary privileges.</li></ol>

Repository Type	Steps
	d. Get the credentials from each of these users, click <b>Username and password</b> , and create an alias for each user name and password.  e. Ask the users to perform step 1, above.
Mirror repositories that were created on other Command Centrals and that are hosted on remote Platform Managers	Click <b>Username and password</b> and create an alias for the user name and password that can access the remote Platform Manager.
Git asset repositories	Click <b>Username and password</b> and create an alias for a Git user name and password (HTTP/S repository URL), or click <b>SSH private key</b> and create and alias for SSH credentials for accessing Git (SSH repository URL).

## Connect to Software AG Repositories

Before connecting to a Software AG repository, make sure that you know the credentials alias of the repository. For details about credentials aliases, see [“Create Credentials Aliases” on page 16](#).

To watch a video that shows how to add repositories and set up Command Central before you start installing products, click <https://www.youtube.com/watch?v=S6LbB5MkZ5k&feature=youtu.be>.

1. Go to **Repositories** and click the tab for the type of repository to which to connect.
2. Click  and do the following:
  - To connect to a product or fix repository, click **Connect to Software AG Repositories**.
  - To connect to an asset repository, click **Connect to Software AG Templates Repository**.
3. Complete the steps in the wizard.

### Related Topics

“Understanding Product and Fix Repositories” on page 30

“Connecting to the Default Templates Repository” on page 75

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

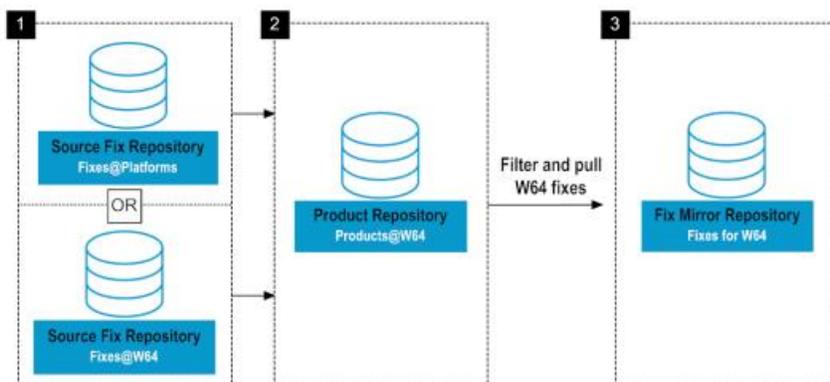
You can watch a demo relating to this task in the Command Central area of the Software AG Tech Community website at <http://techcommunity.softwareag.com>.

If you want to connect to mirror repositories using SSL, the repositories must be hosted on a 10.3 or higher Platform Manager, so you must use Command Central 10.3 or higher. Suppose you have Software AG 9.12 products. You could upgrade to this (latest) release of Command Central. You could then create mirror repositories in the Command Central installation (which always includes a local Platform Manager), or you can install Platform Manager on another machine and create mirror repositories in that installation.

Before Command Central 10.3, when you created a fix mirror repository, you specified one or more source fix repositories, and then you could specify one or more product repositories to filter the source fix repositories by product and release. This filtering reduced the target fix mirror repository to contain only fixes for the products and releases in the source fix repositories. However, Command Central did not filter by operating system, so fixes from every operating system in the source fix repositories were included, and the resulting size of the target fix repository could be quite large. On the Fixes tab, these fix mirror repositories show the value ALL in the OS column.

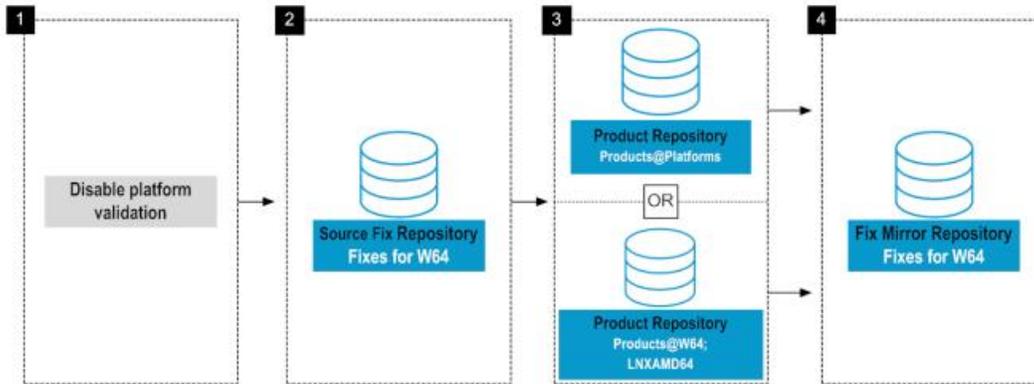
With Command Central 10.3, you can create a fix mirror repository containing fixes for a single operating system by filtering by operating system in addition to filtering by product and release. Use one of the methods below.

- The recommended method is to use source fix repositories that contain the desired fixes for *at least* the desired operating system, and to filter by product, release, and operating system using product repositories that contain products for *only* the desired operating system. For example, to create a fix repository that contains fixes for products on Windows, use source fix repositories that each contain fixes for products on Windows and Linux; or for Windows, Linux, and AIX; or even for all operating systems, but filter by product repositories that are for only Windows.



- This method is the reverse of the previous method. In this method, use source fix repositories that contain fixes for *only* the desired operating system, and filter for product and release using product repositories that contain products for *at least* the desired operating system. For example, to create a fix repository that contains fixes for *product release* on Windows, use source fix repositories that each contain fixes for products on Windows, and filter by product repositories that contain products for Windows; or for Windows and Linux; or for Windows, Linux, and AIX; or even for all operating systems. To use this method, you must first turn off Command Central validation by going to **Environments > ALL > Instances > CCE > Configuration > Java System Properties** and entering this line:

```
com.softwareag.platform.management.client.fix.mirror.platform.validation=false
```



There are two additional requirements when you create a target fix mirror repository for a single operating system:

- You must save the target fix mirror repository on a Command Central 10.3 or later host machine.
- If you are creating the target fix mirror repository for 10.x products, you cannot use fix image repositories created using Update Manager 9.x as source fix repositories. On the Fixes tab, fix image repositories created using Update Manager 9.x show the value UNKNOWN in the OS column. Instead you must use Update Manager 10.x to create any fix image repository you want to use a source fix repository.

On the Fixes tab, fix mirror repositories for a single operating system show the value *operating\_system* in the OS column.

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 **Update Repository Contents**.
4. If you later need to specify different source repositories, click , click **Edit Repository**, and specify the new source repositories.

## Import Product License Keys for Instances or Components

Software AG requires license keys for some products. Each license key contains the license for a product, a product component, or a product feature. Software AG provides license keys when you first license a 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. Add license keys to Command Central as follows:

- a. Open your installation email from Software AG. Attached to the email are individual product license key files, with file names that include product names.
- b. Copy the files to a machine that can access the Command Central from which you will install your products.
- c. Go to **Licensing > Keys**, click , choose the appropriate option, and provide the requested values.

The installation email might also provide license manifest files, each of which contains all contracts for one location of your organization and is used for license metering. License manifest files have file names that do not include any product name. If you have and want to use license manifest files, add them to Command Central by going to **Licensing > Manifests**, clicking , and providing the requested values. Then open each file. If the LicenseManifest Version at the top is 3.0 or higher, license keys are embedded within the file and you need take no further action, because Command Central automatically extracts the keys from the manifest files and lists the keys on the **Keys** tab. If the version is lower than 3.0, proceed to the next step.

**Important:**

In Command Central 10.7, license manifest files and license metering are deprecated without replacement.

2. Assign license keys to product instances or components as follows:
  - a. Go to **Environments > All > Instances** and click an instance or component that requires a license key.
  - b. Click **Configuration**, click **Licenses**, click the license key, click **Edit**, and click the license key to use.
  - c. Repeat for all instances or components that require license keys.

## Register Bootstrap Installers in Command Central

---

When you are adding new product installations in your Command Central landscape, Command Central requires a bootstrap installer to install the Platform Manager instance to manage the installation. The version of the required bootstrap installer is the same as the version of the products in the installation. The operating system of the bootstrapper is the same as the operating system on the machine that hosts the product installation. You can register multiple bootstrap installers in Command Central for different release versions and operating systems.

➤ **To register bootstrap installers in Command Central:**

1. Download the bootstrap installer for the required version and operating system from the Empower Product Support website.

2. Save the bootstrap installer file in *Software AG\_directory* /profiles/CCE/data/installers.

## Additional Configuration for Command Central in a Production Environment

---

### 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.

### Change or Reset the Platform Manager Administrator Password

Software AG strongly recommends that you change the Platform Manager Administrator user password periodically.

#### ➤ To change the Platform Manager Administrator user password:

1. Go to **Environments > ALL > Instances > SPM > Configuration > Internal Users > Administrator** and edit the password.
2. Verify that the password was updated successfully by checking if the status of the Platform Manager instance is online.

## 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.

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.

### Related Topics

“Using the Secure Socket Layer (SSL) Protocol and Certificates in Command Central” on page 499

## Control User Access to Command Central

Restrict access to the Command Central host machine to Command Central users. This is especially important if you are using Command Central to manage production environments.

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**.

## 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.

## 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 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.

The following table describes the monitoring features and their default state.

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. For Command Central, go to **Environments > ALL > Instances > CCE > Configuration**. For Platform Manager, go to **Environments > ALL > Instances > 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**.

## Upgrade Command Central

---

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

- Binary cache.
- Composite templates.
- Credentials.
- Port, SSO, proxy, environment, JAAS, LDAP, and outbound SSL configurations.

- HTTPS, LDAP, and SSO certificates.
- License reports.
- Local mirror and image repositories.
- Users, groups, and roles, and password manager data.
- Any Java system properties that have been customized from the defaults.
- Command Central bootstrap installers.

These configurations and data are migrated for the Platform Manager that is installed with Command Central:

- Port, SSO, and proxy configurations.
- HTTPS and SSO certificates.
- Users, groups, and roles, and password manager data.
- Any Java system properties that have been customized from the defaults.

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 and stop Command Central.
  - 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 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 from the old installation to the new installation. Follow the instructions in the topic in this help for installing Command Central with the additional guidelines below.
  - Specify the argument `-m full_path_to_old_Command_Central_dir_or_ZIP_file`.
  - You must specify the argument `-p password` by providing the administrator user password of the old Command Central. After completing the upgrade, you can change the password by following the steps in [“Change or Reset the Command Central Administrator Password” on page 21](#). Software AG recommends that you change the Command Central administrator user password periodically.
  - 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.

**Important:**

You cannot update the source Platform Manager port configurations during upgrade, since this would break the connection to the migrated mirror repositories. Command Central automatically migrates the port configurations of the source Platform Manager. Even if you provide new Platform Manager ports using the `-s` and `-S` arguments, Command Central will ignore them.

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.

## 2 Creating, Managing, or Upgrading Standalone Product Installations

---

■ Understanding Standalone Product Installations .....	29
■ Understanding Product and Fix Repositories .....	30
■ Understanding Asset Repositories .....	31
■ Default and Custom Product Administrator User Passwords .....	31
■ Connect an Existing Product Installation to Command Central and Add It to Environments .....	32
■ Create a New Standalone Installation .....	33
■ Create and Update Product Instances .....	41
■ Change the Status of an Instance or Component .....	42
■ Install Fixes .....	42
■ Install Support Patches .....	43
■ Compare Products, Fixes, and Instances .....	44
■ View KPIs and Alerts for an Instance or Component .....	44
■ Add an Environment to a Landscape .....	44
■ Configure Shared Secrets to Encrypt Instance and Component Passwords .....	45
■ Encrypt User Passwords for Product Configurations .....	45
■ Group Product Instances or Components .....	46
■ Monitor the Licensing State for a Landscape .....	46

- Upgrade a Standalone Product Installation ..... 46
- Understanding License Metering (DEPRECATED) ..... 48

## Understanding Standalone Product Installations

You can use Command Central to create and upgrade product installations, product instances, and database components within environments. If you want to automate installation and upgrade tasks for product environments, you can develop *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.
- Update existing environments with new fixes, instances, configurations, and files.

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.

**Note:**

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

You can easily use Command Central to monitor and maintain the product instances in your environments, as follows:

- 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.

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

**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.

## 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 directly from Empower on target installation machines.

The connection also enables you to create *mirror* repositories from the repositories on Empower. These mirror repositories reside in your file system and contain the Software AG products or fixes you choose. From these mirror repositories you can 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 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 connect to the Command Central that has Internet access and create secondary mirror repositories from the primary mirror repositories. You 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.

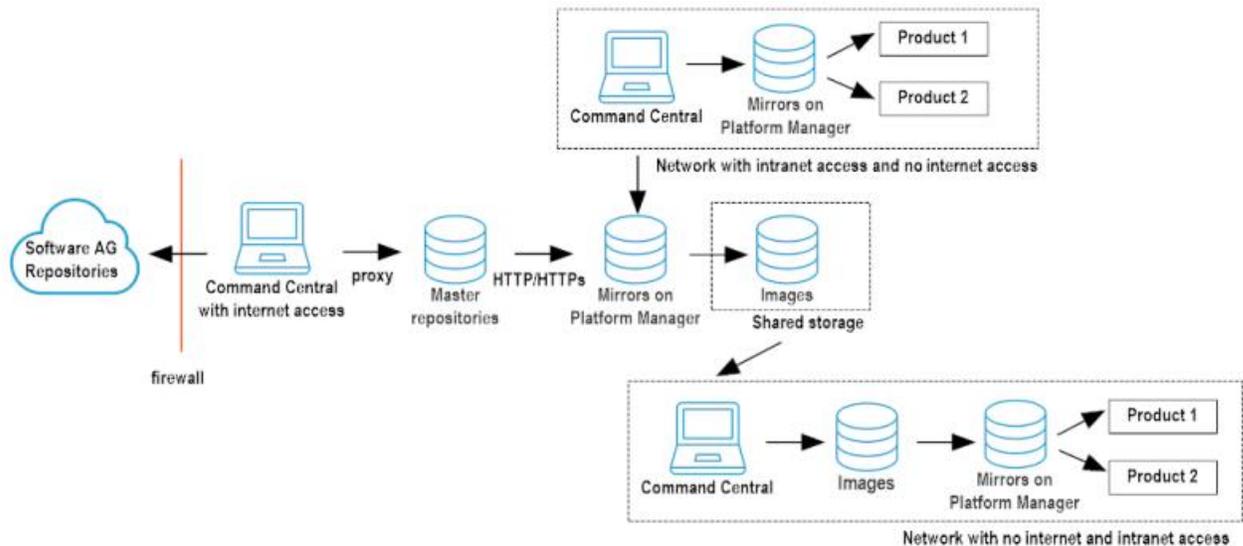
You can refresh an existing mirror repository whenever new products or fixes become available in the source repositories. You can also specify different source repositories from which to populate a mirror repository.

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:**

When you must use image repositories, 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.

The following diagram displays how to Command Central connects to the Software AG repositories and how to use the different types of repositories in a network with intranet access and no internet access and in a network with no internet and intranet access. To watch a video that provides a detailed description of the diagram, click <https://www.youtube.com/watch?v=nks1KF1vKIU>.



## 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, your version control system, and webMethods Asset Build Environment to build assets into composites. Asset Build Environment produces a flat file asset repository consisting of a directory of composites along with related files such as asset composite definition language (ACDL), ZIP, and jar files. You can point to the directory from Command Central, ZIP the directory and upload it to Command Central, or push the composites and related files to a Git repository on a machine accessible to Command Central and then connect Command Central to the Git repository.

## Default and Custom Product Administrator User Passwords

When installing products using Command Central, the initial password for the default administrator user of the product depends on the release version of the product.

### Installing Products with Version 10.11 or Higher

By default, Command Central installs the product using the ADMINISTRATOR credentials alias. Note that the ADMINISTRATOR credentials alias does not have a default password. Before installing products using Command Central, you must edit the ADMINISTRATOR credentials alias by navigating to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials > ADMINISTRATOR** and entering a strong password, or the product installation will fail because the administrator user password is not provided. You can also add

one or more custom credentials aliases for the default administrator users of the products you are installing and specify the custom credentials alias to use when installing a product. Software AG recommends that you set a strong password for each product administrator user. For information about adding credentials aliases, see [“Create Credentials Aliases” on page 16](#).

**Important:**

If you specify a custom credentials alias, Command Central will use the provided password for the default administrator user of the product and ignore the username value of the credentials alias. For example, if you specify a credentials alias that has a username value "my-user" and a password value "my-admin-password" and the default administrator user for the product you are installing is "sysadmin", then Command Central will use the password "my-admin-password" for the "sysadmin" user when installing the product.

## Installing Products with Version 10.7 or Lower

Command Central installs the product and sets the default administrator user password for that product. Software AG recommends that you change the default password as soon as possible following the instructions in [Administering Software AG Products Using Command Central](#) for the required product.

## Connect an Existing Product Installation to Command Central and Add It to Environments

---

1. Go to **Environments > ALL > Installations**, click , click **Add Installation**, and enter the host name of the machine where the product installation is located.
2. In step two of the wizard, if the machine where the product installation is located:
  - Already has a Platform Manager installation, select Platform Manager is already installed, click **Next**, and provide the requested details for connecting to Platform Manager.

**Tip:**

To check the Platform Manager port number, go to the machine, where the product installation is located, navigate to *Software AG\_directory* \profiles\SPM\configuration\com.softwareag.platform.config.propsloader 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.

- Does not have a Platform Manager installation, select **Install Platform Manager**, provide the requested installation details, click **Next**, and select the port number to use for connecting to Platform Manager. For information about installing Platform Manager, see [“Install Platform Manager Using the Command Central Web User Interface” on page 38](#).
3. Complete the remaining steps of the wizard. When you connect Command Central to an existing installation, Command Central automatically adds the installation to **Environments**

> **ALL**. You can also add the installation to one or more specific environments. Go to **Environments > ALL > Installations** and drag and drop the installation onto the target environment in the Environments pane.

**Note:**

If you later remove the installation from every user-defined environment, the installation will still be listed in the ALL environment unless you explicitly remove it from the ALL environment.

## Create a New Standalone Installation

### Create Asset Repositories

Use your continuous integration server, your version control system, and Asset Build Environment to build assets into composites. Asset Build Environment produces a flat file asset repository consisting of a directory of composites along with related files such as asset composite definition language (ACDL), ZIP, and jar files. You can point to the directory from Command Central, ZIP the directory and upload it to Command Central, or push the composites and related 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 Professional 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.
- Software AG requires license keys for some products. Each license key contains the license for a product, a product component, or a product feature. Software AG provides license keys when you first license a 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. Add license keys to Command Central as follows:
  1. Open your installation email from Software AG. Attached to the email are individual product license key files, with file names that include product names.
  2. Copy the files to a machine that can access the Command Central from which you will install your products.

3. Go to **Licensing > Keys**, click , choose the appropriate option, and provide the requested values.

The installation email might also provide license manifest files, each of which contains all contracts for one location of your organization and is used for license metering. License manifest files have file names that do not include any product name. If you have and want to use license manifest files, add them to Command Central by going to **Licensing > Manifests**, clicking , and providing the requested values. Then open each file. If the LicenseManifest Version at the top is 3.0 or higher, license keys are embedded within the file and you need take no further action, because Command Central automatically extracts the keys from the manifest files and lists the keys on the **Keys** tab. If the version is lower than 3.0, proceed to the next step.

**Important:**

In Command Central 10.7, license manifest files and license metering are deprecated without replacement.

- 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

- 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.

- Create fix repositories.

The following table describes the type of repository to create for each product release.

Product Release	Steps
9.9 and later	Create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.
9.8	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.  Create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.
9.7	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.  Create image repositories that contain the fixes to install using the instructions in this help.

- 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.

## 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, the Database Component Configurator, and the database scripts for your products. Install the Database Component Configurator and the database scripts on a machine that has access to your database server and then install the latest fixes on the Database Component Configurator.

For information on supported RDBMSs, see *System Requirements for Software AG Products*.

For information about database connections, database drivers, database components that are required by each product, and tasks to perform before creating database components, see *Installing Software AG Products*.

## Install and Configure Platform Manager for a New Standalone Installation

### 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. For information

about configuring OpenSSH on Windows machines using Cygwin, see [Using Cygwin to Configure OpenSSH When Installing Platform Manager on a Remote Windows Machine](#) on the Software AG Tech Community website.

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

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

1. Log on to the target machine as the installation user you created when you installed Command Central.
2. Follow the instructions in your installation email from Software AG to download the Command Central bootstrapper for your operating system.

**Note:**

Software AG refreshes the bootstrapper with new fixes. Always use the latest bootstrapper to install Platform Manager.

**Note:**

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

3. Launch a command shell. On Windows, launch using Run as Administrator.
4. Run the .bat file or .sh script. The basic command using default values is shown below, where `--accept-license` means you accept the Software AG product license agreement (available for review at [http://documentation.softwareag.com/legal/general\\_license.txt](http://documentation.softwareag.com/legal/general_license.txt)).

```
cc-def-release-fixnumber-operating_system.{bat|sh} -D SPM --accept-license
```

The following table shows arguments you can specify on the command to override the default values.

Argument	Value	Default
<code>-H host_name</code>	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
<code>-d path</code>	Full path to the directory in which to install Platform Manager.	Directory from which you are running the bootstrapper

Argument	Value	Default
-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 manage user account. If you are installing in a production environment, Software AG recommends you provide a strong, user-defined password.	
-P <i>root_password</i>	On UNIX, sudo password to use to register a daemon for Platform Manager.	

**Note:**  
You can register daemons at a later time by re-running the bootstrapper with this argument.

## 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-fix3-lnxamd64.sh -d /opt/softwareag -D SPM
-H linuxhost1.com -s 9997 -S 9998 -p manage456 -P superuser123
```

```
solarishost2>./cc-def-9.12-fix3-solamd64.sh -d /opt/softwareag -D SPM
-H solarishost2.com -s 9997 -S 9998 -p manage456 -P superuser890
```

```
windowshost3>cc-def-10.7-fix1-w64.bat -d C:\softwareag -D SPM
-H windowshost3.com -s 9997 -S 9998 -P manage456
```

## Install Platform Manager Using the Command Central Web User Interface

You can use the Command Central web user interface to create a new Platform Manager installation:

- On a remote machine.
- On the local machine where Command Central is installed, in a different directory.

In the Command Central web user interface, you access the **Add Installation** wizard by navigating to **Environments > ALL > Installations** and clicking the plus icon. You should specify the name of the machine where to install Platform Manager. To install Platform Manager, in step two of the wizard select **Install Platform Manager** and provide the requested values.

The bootstrap installer you select must be for the same version as the products you are going to manage with the Platform Manager installation, and for the operating system of the machine you entered in step one of the wizard. If the bootstrapper you require is not listed in the drop-down

box, add it in Command Central by following the steps in [“Register Bootstrap Installers in Command Central” on page 20](#).

By default, Command Central uses the ADMINISTRATOR credentials alias for the Platform Manager administrator user. Note that the ADMINISTRATOR credentials alias does not have a default password and you must edit the alias and enter a strong password before using it to install Platform Manager. To edit the ADMINISTRATOR credentials alias, navigate to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials**.

Optionally, you can select a different credentials alias for the Platform Manager administrator user, depending on the Platform Manager version you are installing. If you are installing Platform Manager version:

- 10.11 or higher, you can use a custom credentials alias that you have created in advance.
- 10.7 or lower, you can use the default administrator user password, provided in the DEFAULT\_ADMINISTRATOR credentials alias, or a custom credentials alias you have created in advance.

**Important:**

If you specify a custom credentials alias, Command Central will set the provided password for the Platform Manager administrator and ignore the username value of the credentials alias. For example, if you specify a credentials alias that has a username value "my-user" and a password value "my-admin-password", Command Central will set the password "my-admin-password" to the "Administrator" user when installing the product. For information about creating credentials aliases, see [“Create Credentials Aliases” on page 16](#).

For information about administrator user passwords for different release versions, see [“Default and Custom Product Administrator User Passwords” on page 31](#).

If you are installing Platform Manager on a remote machine, you can optionally provide the following advanced configurations after clicking the **Advanced** button on the first page of the Add Installation wizard:

- Substitute user credentials for the user that installs and starts Platform Manager. If you do not specify substitute user credentials, Command Central installs and starts Platform Manager using the specified SSH credentials. For more information about the substitute user, see [“Bootstrapping on a Remote Machine with a Substitute User” on page 146](#).
- SSH port number for the remote machine. If you do not specify a remote SSH port, Command Central uses port 22.

You can install Platform Manager into multiple directories at the same time. For directories that are on the same machine, the Platform Managers 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.

## Supported Key Exchange Methods for SSH Connections

By default, Command Central 10.5 and higher uses strong key exchange methods that meet the current security requirements for SSH connections. Command Central allows the following key exchange methods (listed in order of priority): diffie-hellman-group14-sha256 (highest),

diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, diffie-hellman-group14-sha1 (lowest).

Only when required by the environment, you can also configure Command Central to allow the following key exchange methods, which are disabled by default, because they are weak:

- diffie-hellman-group1-sha1
- diffie-hellman-group-exchange-sha1

You can use the `com.softwareag.platform.management.client.remoteaccess.allowedkexmethods` property to control the list of key exchange methods allowed by Command Central. For details about the property, see [“SSH Connections” on page 497](#).

If the OpenSSH server on the remote machine does not use any of the key exchange methods allowed by Command Central, you can configure the OpenSSH server to use one or more of the allowed methods. For more information, see [Configuring an OpenSSH Server to Use the Key-Exchange Methods Allowed by Command Central](#).

## 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.

## Create a New Standalone Product Installation and Add It to Environments

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.

**Note:**

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

3. If you are installing products with version 10.11 or higher, select the credentials alias to use for the administrator user password of the products. For more information, see [“Default and Custom Product Administrator User Passwords”](#) on page 31.
4. After installation is complete, you can add it to one or more environments. Go to **Environments > ALL > Installations** and drag and drop the installation onto the target environment in the Environments pane.

**Note:**

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

5. Install the latest fixes on the installed products using the instructions in this help.
6. 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.

7. If one or more of your products require a license key, see [“Import Product License Keys for Instances or Components”](#) on page 19.
8. 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](#).
9. Create database components using the instructions for the [“sagcc exec administration”](#) on page 222 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.

## Create and Update Product Instances

For instructions on creating and updating product instances, see [Administering Software AG Products Using Command Central](#). You can also set up the product configurations listed in [Software AG Command Central Feature Support Matrix](#).

## 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.

The following table explains how to download one, multiple, or all logs (for example, because you want to compare them, or use tools to process them).

Logs to Download Steps	
One	Click  for that log.
Multiple	Hold down the Shift or Ctrl key and select the log rows. Click  and then click <b>Download selected logs</b> .
All	Click <b>Download selected logs</b> without selecting any log.

## 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 Tech Community website at <http://techcommunity.softwareag.com>.

If any support patches exist on the target products or components, Command Central automatically uninstalls them before installing the fix. 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 [“Before Installing Fixes and Support Patches”](#) on page 35.
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.

The table below lists the operating system name for each value that can appear in the OS column.

Value	Operating System
AIX	IBM AIX
OSX	Apple Mac OS X
HP11IT	HP HP-UX Intel Itanium 2
HP11	HP HP-UX PA-RISC
LNXS390X	Linux RHEL and SLES IBM System z
LNX	Linux RHEL and SLES x86
LNXAMD64	Linux RHEL and SLES x86-64 (EM64T, AMD64)
WNT	Microsoft Windows x86
W64	Microsoft Windows x86-64 (EM64T, AMD64)
SOL	Oracle Solaris SPARC
SOLAMD64	Oracle Solaris x86-64 (EM64T, AMD Opteron)

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 [“Before Installing Fixes and Support Patches”](#) on page 35.
2. Go to **Environments > ALL > Installations** and click 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 .

## 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.

## 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.

## 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**.

## 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.

## Encrypt User Passwords for Product Configurations

---

You can encrypt user passwords for product configurations and use the encrypted values in place of clear-text passwords in templates, properties files, and so on. If the installation that contains the products belongs to a single environment, and you configured a shared secret for that environment, Command Central uses the environment shared secret to encrypt the password. Otherwise, Command Central uses the global shared secret to encrypt.

Software AG recommends using the “[sagcc exec security encrypt](#)” on page 420 command to encrypt passwords. However, you can use the steps below to encrypt passwords if you want to use the encrypted password in a template or template properties file that provisions the same configuration. The steps explain how to export an encrypted password for a configuration instance.

1. Go to **Environments > ALL > Instances > *instance\_name***.
2. On the Configuration tab, click a configuration type that includes a password, then click the alias for a configuration instance.

3. Click **Export** and copy the encrypted value for the password property from the template snippet.

## Group Product Instances or Components

---

You can tag product 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. On the Overview tab, add the attributes.

## Monitor the Licensing State for a Landscape

---

If you want to check the current state of licensing for installed products in a Command Central landscape, you can generate an installation report. An installation report contains detailed information for each installed product such as the type of product installation (for example, development or production), 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 units in use (for example, processor cores and named users or events) and the total number of CPU cores per environment. Go to **Licensing > Reports**, click , and click **Create installation report**.

## Upgrade a Standalone Product Installation

---

To see which products support upgrade using Command Central, see the [Software AG Command Central Feature Support Matrix](#).

If you want to upgrade products in one environment using Command Central, follow the instructions in *Upgrading Software AG Products On Premises*.

If you want to automate product upgrade for multiple similar environments using Command Central, see the composite template instructions in this help. You can reconfigure endpoints such as host names and ports by adding actions in the templates. You will still need to perform the manual tasks in [Upgrading Software AG Products On Premises](#) if you cannot script them.

If you do not want to automate the upgrade of Platform Manager, Command Central bootstrapper 10.7 or later enables you to upgrade Platform Manager manually, separately from the products. The bootstrapper migrates these Platform Manager configurations and data:

- Port, SSO, proxy, and JAAS configurations.
- HTTPS and SSO certificates.
- Users, groups, and roles, and password manager data.
- Any Java system properties that have been customized from the defaults.

Follow the instructions below to upgrade Platform Manager manually.

1. If you are going to install the new Platform Manager on a different machine than the old Platform Manager, create a ZIP file of the old Platform Manager. These instructions use the Java Archive tool to create the ZIP file.

- a. Go to the old Platform Manager machine and stop Platform Manager.
- 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 Platform Manager.
- d. If you want to reduce the size of the ZIP file, move the log files out of the `old_Software AG_directory\SPM\logs` directory.
- e. Enter this command:

```
jar cfm ZIP_file common\conf profiles\SPM install\products
```

- f. Copy the ZIP file to any directory on the new Platform Manager 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. Run the Command Central bootstrapper with the arguments below, where `--accept-license` means you accept the Software AG product license agreement (available for review at [https://documentation.softwareag.com/legal/general\\_license.txt](https://documentation.softwareag.com/legal/general_license.txt)).

```
cc-def-release-fixnumber-operating_system.{exe|sh} -D SPM
-m full_path_to_old_Platform_Manager_dir_or_ZIP_file
-d full_path_to_new_Platform_Manager_dir
-s old_Platform_Manager_HTTP_port -S old_Platform_Manager_HTTPs_port
--accept-license
```

**Note:**

If for some reason you have to specify ports that are different from the old ports, Command Central will disable the specified ports after the upgrade and will communicate with the new Platform Manager using the ports used by the old Platform Manager.

## Examples

- To migrate the Platform Manager configurations from C:\sag105\cc\PlatformManager to C:\sag107\cc\PlatformManager using HTTP port 8172 and HTTPS port 8173:

```
cc-def-10.7-fix1-w64.bat -D SPM -d C:\sag107\cc\PlatformManager -m  
C:\sag105\cc\PlatformManager  
-s 8172 -S 8173 --accept-license
```

## Understanding License Metering (DEPRECATED)

---

### Deprecation Impacts

In Command Central 10.7, license metering and license manifest files are deprecated without replacement. For monitoring the current state of licensing, use installation reports as described in [“Monitor the Licensing State for a Landscape”](#) on page 46.

### License Manifests (Deprecated)

License metering depends on license manifest files. Each license manifest file issued by Software AG contains all contracts for one location of your organization. If you received license manifest files from Software AG, you can generate or view the reports described in this topic.

### Current State of Licensing for a Landscape (Deprecated)

If you want to check the current state of licensing for installed products in a Command Central landscape, you can generate a *snapshot report*. A snapshot report contains detailed information for each installed product such as the type of product installation (for example, development or production), 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 units in use (for example, processor cores and named users or events) and compares that number with the number of licensed units specified in the license manifest for that location. The type of unit being metered depends on the licensed metric (for example, processor cores for metric PCB and events for metric EVT). A summary indicates whether your landscape matches the available licenses, and lists the number of products that match or do not match the licenses. Go to **Licensing > Reports**, click , and click **Create license manifest report**.

# 3 Creating or Upgrading Software Stacks

---

- Understanding Software Stacks ..... 50
- Understanding Product and Fix Repositories ..... 51
- Create a Software Stack ..... 52
- Create Software Stacks from Existing Environments ..... 57
- Rename a Software Stack or Layer ..... 58
- Upgrade Software Stacks ..... 58

## 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*.

An additional benefit of using stacks is to simplify future upgrades, as the upgrade procedure for stacks is simpler than the procedure for standalone product installations.

**Note:**

The new stacks feature is available as an early iteration for you to investigate. Stacks you create cannot yet be updated. 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.

- The Infrastructure layer type contains Platform Manager. The template identifies the physical machines, virtual machines, or Docker containers that host Platform Manager.
- The Runtime layer type contains instances of one type of runtime. The template specifies instance creation, configuration, and licensing. The runtime instances in one stack cannot be used in any other stack.
- The Database layer contains the database components you are using with the runtimes. The template specifies the database name, the location of the Database Component Configurator, and the database connection parameters.

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.

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.

- Compare instance configurations, products, and fixes.

You can also upgrade the runtime instances in software stacks. For more information about upgrading products using Command Central stacks, see *Upgrading Software AG Products On Premises*.

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.

To watch a video that provides a detailed description of the stacks and layers concepts in Command Central, click <https://www.youtube.com/watch?v=qPyNpOzzrjA>.

## 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 directly from Empower on target installation machines.

The connection also enables you to create *mirror* repositories from the repositories on Empower. These mirror repositories reside in your file system and contain the Software AG products or fixes you choose. From these mirror repositories you can 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 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 connect to the Command Central that has Internet access and create secondary mirror repositories from the primary mirror repositories. You 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.

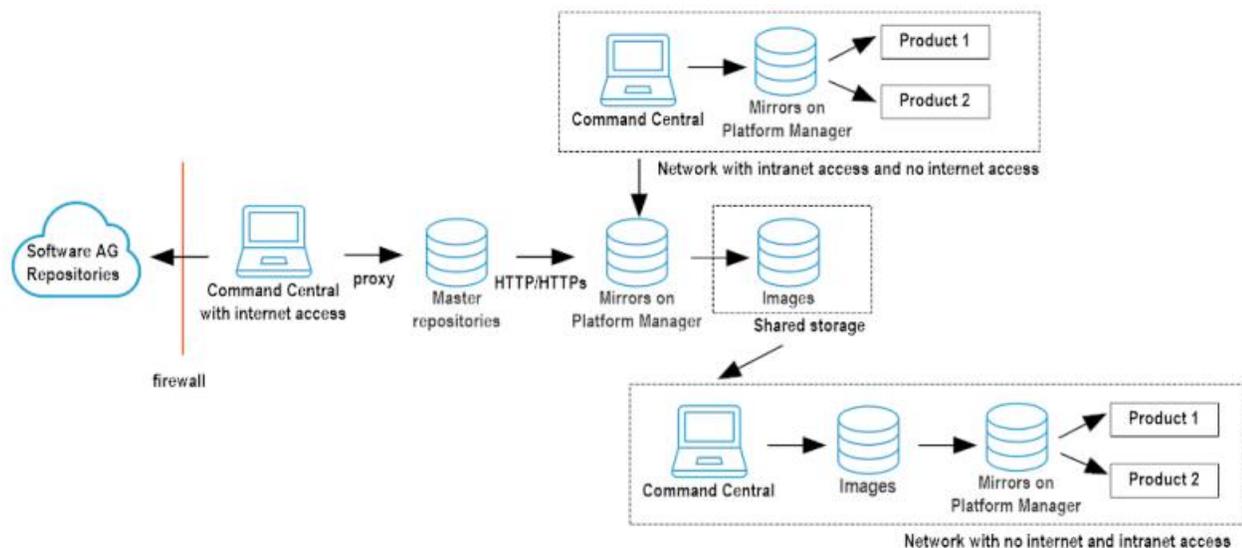
You can refresh an existing mirror repository whenever new products or fixes become available in the source repositories. You can also specify different source repositories from which to populate a mirror repository.

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:**

When you must use image repositories, 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.

The following diagram displays how to Command Central connects to the Software AG repositories and how to use the different types of repositories in a network with intranet access and no internet access and in a network with no internet and intranet access. To watch a video that provides a detailed description of the diagram, click <https://www.youtube.com/watch?v=nks1KF1vKIU>.



## Create a Software Stack

### Before Installing Products

- Work with your administrators, Software AG Professional 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.
- Software AG requires license keys for some products. Each license key contains the license for a product, a product component, or a product feature. Software AG provides license keys when you first license a 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. Add license keys to Command Central as follows:
  1. Open your installation email from Software AG. Attached to the email are individual product license key files, with file names that include product names.
  2. Copy the files to a machine that can access the Command Central from which you will install your products.
  3. Go to **Licensing > Keys**, click , choose the appropriate option, and provide the requested values.

The installation email might also provide license manifest files, each of which contains all contracts for one location of your organization and is used for license metering. License manifest files have file names that do not include any product name. If you have and want to use license manifest files, add them to Command Central by going to **Licensing > Manifests**, clicking , and providing the requested values. Then open each file. If the LicenseManifest Version at the top is 3.0 or higher, license keys are embedded within the file and you need take no further action, because Command Central automatically extracts the keys from the manifest files and lists the keys on the **Keys** tab. If the version is lower than 3.0, proceed to the next step.

**Important:**

In Command Central 10.7, license manifest files and license metering are deprecated without replacement.

- 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

- 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.

- Create fix repositories.

The following table describes the type of repository to create for each product release.

Product Release	Steps
9.9 and later	Create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.
9.8	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.  Create local mirror repositories that contain the fixes and support patches to install using the instructions in this help.
9.7	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.  Create image repositories that contain the fixes to install using the instructions in this help.

- 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.

## 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, the Database Component Configurator, and the database scripts for your products. Install the Database Component Configurator and the database scripts on a machine that has access to your database server and then install the latest fixes on the Database Component Configurator.

For information on supported RDBMSs, see *System Requirements for Software AG Products*.

For information about database connections, database drivers, database components that are required by each product, and tasks to perform before creating database components, see *Installing Software AG Products*.

## Create a Software Stack

To watch a video that shows how to create a stack with layers and use the stack to create product installations, click <https://www.youtube.com/watch?v=spAZk9SUE-c&feature=youtu.be>.

1. Install Platform Manager on target nodes.
  - Command Central can automatically install Platform Manager automatically in the new product installation. To do so, Command Central uses the Command Central bootstrapper. Follow the instructions in your installation email from Software AG to download the bootstrapper of the new release for your operating system, and then store the bootstrapper in the *Command Central\_directory/profiles/CCE/data/installers* directory.
  - If the user account created to install products does not allow remote access from Command Central, you will need to manually install Platform Manager on target nodes using the instructions on installing Platform Manager using the Command Central bootstrapper in this help. 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 aliases for 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 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.

**Note:**

Note that Command Central supports only the PKCS #1 definition of the ASN.1 syntax for representing the SSH private keys.

3. Develop and register micro templates for layers that install products, install fixes, and create database components. 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. See the micro template instructions in this help.
4. Create definitions of layers to use in stacks using the methods below. At minimum, 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. For instructions and locations, see instructions on connecting to the default templates repository in this help.
  - Go to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Layer Definitions**, click , and provide the requested values.
5. Go to **Stacks**, hover over the , and click **Create stack from scratch**. Add the stack and then add layers. At the end of the wizard, provisioning begins automatically.
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*.
7. Connect the products to their database components using the instructions in the installation guide for your products.
8. For additional actions you can take with product installations, see the topics for managing existing standalone product installations in this help.

## Create Software Stacks from Existing Environments

You can create software stacks from existing installations. To do so, you will identify installations to Command Central, and Command Central will create one stack for each set of installations that are the same release, are installed on the same operating system and in a directory of the same name, and whose Platform Manager uses the same port and protocol (HTTP or HTTPS).

To watch a video that provides a step-by-step demonstration on how to create stacks from existing environments in Command Central, click <https://www.youtube.com/watch?v=GomeIhvtA3I>.

1. Go to **Environments**, click the environment that contains the installations from which to create stacks, click  at the bottom right of the **Environments** pane, and then click **Auto-create stacks**.
2. In the **Auto-create stacks** dialog box, in the **Product releases** drop down, click the release of the installations from which to create stacks or click **ALL**.

By default, Command Central will name the stacks 1, 2, 3, and so on. In the **Stack prefix** field, specify a prefix for those names that will help you better identify the stacks later.

3. Click **OK** to create the stacks.
4. If you need a database layer in a stack, add it manually.

## Rename a Software Stack or Layer

---

If you want to rename a software stack or layer, see the [“sagcc update stacks alias” on page 449](#) and [“sagcc update stacks layers alias” on page 449](#) commands in this help.

## Upgrade Software Stacks

---

The upgrade procedure for stacks is simpler than the procedure for standalone product installations. For stacks, Command Central automatically generates composite templates that you can then use to upgrade a single environment or to automate the upgrade of multiple environments. If you are not yet using stacks, you can create stacks from the installations you want to upgrade. For instructions on all of these topics, see *Upgrading Software AG Products On Premises*.

To see which products support upgrade using Command Central, see the [Software AG Command Central Feature Support Matrix](#).

# 4 Common Use Cases with Command Central

---

- How to Get Templates for Upgrade ..... 60
- Install or Upgrade Product Installations with Composite Templates ..... 63
- Install or Upgrade Product Installations with Stacks ..... 67
- Update Licenses on a Number of Nodes with Templates ..... 69

## How to Get Templates for Upgrade

---

After you complete the steps in this topic, you will have templates that you can use to upgrade the Software AG product installations that you manage through Command Central.

### Prepare for the Upgrade Templates

- Choose one of the supported upgrade methods described in *Upgrading Software AG Products On Premises* and check in the same guide for product-specific upgrade requirements.
- Check the [Software AG Command Central Feature Support Matrix](#) for the target migration release versions and migration types, supported for the products in the installation.

You can choose one of the following options (based on whether you already use composite templates and the release version of the source product installations):

Option 1: Use the same composite templates you used to provision the source installations

Option 2: Generate templates from the source installations

Option 3: Create stacks for upgrade (from scratch or with auto-create stacks)

### Level of Difficulty

Moderate for options 1 and 2. If you already have composite templates for provisioning, you only invest in modifying them for the upgrade. If you do not use composite templates, option 2 is better for large environments with custom configurations.

Low for option 3. This option works best for standard product installations.

### Option 1: Use the same composite templates you used to provision the source installations

If you installed the source product installations using composite templates, you can use the same templates for the upgrade. Note that when you want to use a composite template for upgrade, the template must include the [“Provision” on page 102](#) section with the target nodes, on which to create the target installations. If you use `overinstall` as the upgrade method, you must also include the [“Migration” on page 97](#) section in the template to describe the source nodes.

The default behavior of Command Central during the upgrade is described in the *Software AG Command Central Help* (in the [how Command Central processes a composite template for installations of release “10.3 and higher” on page 116](#) and [“10.1 and lower” on page 119](#) topics). You can change the default upgrade behavior by editing the following sections in the composite template:

- Specify custom migration settings for the installations that you want to upgrade in the "migration:" section as described in [“Migration” on page 97](#)
- Define the product instance properties that you want to migrate as described in [“Migrating a Product Instance” on page 125](#)

- If you want to update the composite templates with configuration settings you applied after migrating the installations, you can export a single product configuration and add it in the template. See [“Adding Configurations in an Exported Micro Template” on page 81](#).
- If you want to migrate to a different host using a cloned database or disconnected migration, modify the composite template as described in:
  - [“Migrating to a Different Host with Cloned Database” on page 122](#)
  - [“Migrating Using Disconnected Migration” on page 124](#)
- When migrating product installations to a different host, you must also change the hosts in the end-point configurations of the products to point to the new hosts.

**Important:**

Do not modify the template files located under the Command Central profile. Store the composite templates that you want to modify and prepare for the upgrade under version control and re-import them into Command Central when you want to run the upgrade.

**Option 2: Generate templates from the source installations**

With this option, you create the templates for the upgrade by generating templates from the source product installations. Command Central exports the details about the Platform Manager of the source node, the product instances installed on the source node, and the database configured for the source node into the generated templates. You can then customize the generated templates if required.

**Important:**

The sections and parameters included in the generated templates will depend on the release version of the Platform Manager installation from which you export the templates.

1. Generate the following templates from the source product installations:
  - A migration template for the Platform Manager of the source installation [“Exporting the Platform Manager of an Installation to a Template” on page 77](#) and [“sagcc exec templates composite generate migration” on page 472](#)
  - Product instance templates for migration "Usage Notes for Exporting an Instance to a Migration Template in [“sagcc exec templates composite generate” on page 457](#)

Note that when generating a template for upgrade, you do not include the product configurations, because during the template application Command Central invokes the product migration utilities, which migrate the configuration settings from the source product instance. However, when you migrate to a different host, you can export the end-point configurations for the product instances and then edit the generated template to update the hosts in the end-points to the new hosts.

If you export product instances without adding the `options=INFRASTRUCTURE` argument (in the [“sagcc exec templates composite generate” on page 457](#) command), the generated templates do not include a `nodes:` section and you must add this section in the generated templates.

- Database template for the database of the source installation (supported for installations of release version 10.5 and higher) [“Exporting the Database for Products in an Installation to a Template” on page 81](#) and [“sagcc exec templates composite generate” on page 457](#)

Note that the database template that gets generated for a node does not include a `nodes:` section and you must add this section in the generated template.

The generated templates are imported in Command Central and get listed in the Templates view.

2. When the generated product instance and database templates do not have a `nodes:` section, which is required for the upgrade, choose one of the following:
  - Merge the templates generated for the Platform Manager, product instances, and database of the node into one composite template.
  - Copy the `nodes:` section from the template generated for the Platform Manager of the node to the database template and each product instance template generated for the node. When you run the upgrade using the generated templates, you must always run the generated Platform Manager template first, and then run the database and product instance templates using the same parameters that you specified in the `nodes:` section of the Platform Manager template.

**Important:**

Do not modify the template files located under the Command Central profile. Store the templates that you want to modify and prepare for the upgrade under version control and re-import them into Command Central when you want to run the upgrade.

### Option 3: Create stacks for upgrade (from scratch or with auto-create stacks)

You can create a product stack, in which you add logical layers that reference templates, and use this stack to install or upgrade product installations. If you create a stack and add layers in the stack that reference templates, such as the default templates or templates generated from existing installations, we say that you create the stack from scratch. If you use the auto-create stacks function, Command Central automatically creates the stacks (and their layers) from existing installations that are standard and managed by Command Central.

With auto-create stacks, you can filter the set of installations to add in the stacks by release version or by environment. Command Central generates composite templates for the product installations that you specify and organizes the templates in stacks and layers. For details about auto-creating stacks for upgrade, see [“Upgrade Software Stacks” on page 58](#) and [“sagcc create stacks auto” on page 426](#).

The pros of using stacks to upgrade the product installations is that you can use the Stacks view of the Command Central web user interface for all upgrade-related tasks and you do not require template development knowledge.

## Install or Upgrade Product Installations with Composite Templates

---

After you complete the steps in this topic, you will have a set of Software AG products installed on several remote machines. You can use the steps to install fresh installations or upgrade existing installations to a higher release version. The products in the fresh or upgraded installations will have product instances, which are configured with the settings you define in the composite template(s).

Use composite templates when you want to provision large product environments (with lots of installations) in which you have:

- Custom adjustments that are not standard across the product installations
- Product configurations that are not managed through Command Central

### Level of Difficulty

High. Composite templates are a complex topic that requires significant experience with the webMethods suite. They represent infrastructure as code, but require knowledge on how the products interact and work together. With that in mind, the composite templates are a powerful tool that can automate a huge number of tasks over a big deployment.

### Prepare for the Install or Upgrade with Composite Templates

- It takes some time to learn and become productive with composite templates. If you have DevOps experience, that will help you progress faster.
- Install and configure Command Central as described in the *Software AG Command Central Help*.
- Make sure you have valid license keys for the Software AG products that you include in the composite templates.
- Check the [Software AG Command Central Supported Features Matrix](#) for the supported product configuration and migration types.
- When installing products, check *Installing Software AG Products* for any product-specific requirements.
- When upgrading products, follow the instructions in *Upgrading Software AG Products On Premises* to prepare the existing installations for the upgrade.
- Download the Command Central bootstrap installer (for the release version of the fresh installation or the target upgrade version) and save the downloaded Command Central bootstrap installer file into the `$CC_HOME/profiles/CCE/data/installers` directory. You can then add this bootstrap installer in the “nodes:” section of the composite templates.

## Step 1: Set up Repositories

The repositories are the source of the product or fix installation files. You can connect to the Software AG repositories (hosted on Empower and added in Command Central as master repositories), create a local copy of the Empower repositories (added as mirror repositories), or use a product or fix image (created by the Software AG Installer or Software AG Update Manager and added as an image repository). Command Central will use the repositories to install the products and apply the fixes available for those products.

1. Choose a repository setup that works for you: [“Understanding Product and Fix Repositories” on page 30](#).
2. Set up the repository credentials:
  - [“Create Credentials Aliases” on page 16](#) (the web user interface)
  - "COMMON-CREDENTIALS Usage Notes" in [“Configuration Types for Command Central and Platform Manager OSGI ENGINE” on page 487](#) (the command line interface)
3. Add the repositories:
  - [“Connect to Software AG Repositories” on page 17](#); [“Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes” on page 17](#) (the web user interface)
  - [“sagcc add repository products” on page 390](#); [“sagcc add repository fixes” on page 386](#) (the command line interface)

## Step 2: Import the Product License Keys in Command Central

Import the license keys for the products that you want to install (or upgrade) and assign an alias to each license key: [“Import Product License Keys for Instances or Components” on page 19](#) and [“sagcc add license-tools keys” on page 325](#).

## Step 3: Configure the SSH Connection to Each Remote Machine

Check if the UNIX or Windows operating system on the remote machine has a Secure Shell (SSH) server running or WinRM support (for Windows only). Also, make sure that the system is configured for remote access with the user account for the Software AG products.

- Set up the connection to the remote machine and the credentials alias for the SSH user: [“Bootstrapping Platform Manager Using the Default SSH and WinRM Templates” on page 146](#); "COMMON-CREDENTIALS Usage Notes" in [“Configuration Types for Command Central and Platform Manager OSGI ENGINE” on page 487](#)
- Optional. Configure a substitute user: [“Bootstrapping on a Remote Machine with a Substitute User” on page 146](#)

## Step 4: Prepare the Composite Template

You can design and develop composite templates from the default templates in the [sagdevops-templates](#) project or from templates generated from existing installations ([“sagcc exec templates](#)

[composite generate” on page 457](#)). You can also design and develop custom templates on your own.

### 1. Choose how to design the composite templates.

#### ■ Design Templates by Product

Typically, in a Software AG installation you have base products (such as Integration Server) and products that depend on the base products (such as Trading Networks). You have the following options:

- Create a composite template for the base product(s) in the installation and define different environments in the template. For example, create one composite template for Integration Server (IS) and another composite template for Universal Messaging (UM). In the IS composite template, define an environment type for each IS installation that you want to create. For example, define one environment type for IS with the Adapter for JDBC, and another environment type for IS with Trading Networks.

Pros: With this approach, it is easier to run an installation or upgrade, because the order in which the products get installed and configured is handled inside the template.

Cons: The composite template(s) could get long and complex, which makes them harder to maintain.

- Create one composite template for the base product and another template for the dependent products. For example, create one template for IS and a separate template for Trading Networks (TN). When applying the templates, apply the IS template first and the TN template second.

Pros: The composite templates are smaller and not so complex, which makes them easier to read and maintain.

Cons: When applying the composite templates, you should follow specific steps, such as the specific order, in which to apply the templates. This approach leaves you open to errors.

#### ■ Design Templates by Goal

You can design the composite templates around the intended goal of the template (provisioning or upgrade). The supported upgrade method will also influence the template design. The following list gives you the recommended design options with their pros and cons.

- Create one composite template for provisioning and upgrade. This option is recommended when running cross-host upgrade.

Pros: All the data is in one template and you do not have duplicate sections and properties in the template.

Cons: The result is a very long template with data that is not required when using the composite template for upgrade.

- Create two composite templates: one template contains the product and the other template contains the product configurations. Use both templates for provisioning:

apply the product template first and the product configuration template second. When running a same host (overinstall or side-by-side) upgrade, use only the product template.

Pros: The product template has only the data you require for the upgrade.

Cons: You have some duplicate sections and properties across the two templates. To prevent the product from starting during the execution of the product template, you must set the parameters in the "options:" section of the product template. See ["Options" on page 86](#).

Note that if you use two composite templates (base and configuration) for cross-host migration, the base product template should include the configurations that change the host names.

## 2. Write the composite templates.

Use the following topics as reference when writing the template YAML file:

- ["Understanding the Composite Template Definition" on page 85](#)
- ["How Command Central Processes a Composite Template" "10.3 or higher" on page 116; "10.1 or lower" on page 119](#)
- Optional. Add a database layer in the product composite template: ["Managing Database Components Using a Template" on page 152](#)

## Step 5: Prepare the property files to use with the composite template(s)

Create one or more property files with the values for the input parameters defined in the "environments:env.type:" section of the composite template(s). Property files help you specify different values for the different environments (defined in a template) without editing the template YAML file directly.

## Step 6: Apply the Composite Templates

Import the composite templates in Command Central with the ["sagcc exec templates composite import" on page 454](#) CLI command or from the Templates view in the web user interface.

If you have more than one composite template, apply each template in turn with the ["sagcc exec templates composite apply" on page 451](#) CLI command. The order of applying the templates should go:

1. Base product templates
2. Dependent product templates
3. Product configuration templates

Command Central automatically validates a template after you run the apply composite templates command: ["Validating a Composite Template" on page 129](#).

## Step 7: Monitor and Troubleshoot the Template Application

Use the Jobs view in the Command Central web user interface to monitor the status of the template apply job and access the logs: [“Correcting a Failed Composite Template Apply Operation”](#) on page 130.

## Install or Upgrade Product Installations with Stacks

After you complete the steps in this topic, you will have installed or upgraded the Software AG product installations in your environment through Command Central stacks.

A stack represents a set of product installations. In the stack, the products from the installations are organized in logical layers, see [“Understanding Software Stacks”](#) on page 50. The layers in the stack refer to Command Central templates, which have the details required to install or upgrade the products. The templates are imported in Command Central and you can define layers in different stacks that refer to the same template. Based on how you run the stack, you can create fresh installations or upgrade the existing installations that Command Central manages.

### Level of Difficulty

Moderate to Low. When you work with the Command Central stacks, most of the install or upgrade actions are automated, but you should be familiar with how the products in the webMethods product suite interact and work together.

### Prepare for the Install or Upgrade with Stacks

- Install and configure Command Central as described in the *Software AG Command Central Help*.
- Make sure you have valid license keys for the Software AG products that you include in the stacks.
- Check the [Software AG Command Central Supported Features Matrix](#) for the supported product configuration and migration types.
- When installing products, check *Installing Software AG Products* for any product-specific requirements.
- When upgrading products, follow the instructions in *Upgrading Software AG Products On Premises* to prepare the existing installations for the upgrade.
- Download the Command Central bootstrap installer (for the release version of the fresh installation or the target upgrade version) and save the downloaded Command Central bootstrap installer file into the `$CC_HOME/profiles/CCE/data/installers` directory. You can then specify this bootstrap installer in the infrastructure layer of the stack.

### Step 1: Set up Repositories

The repositories are the source of the product or fix installation files. You can connect to the Software AG repositories (hosted on Empower and added in Command Central as master repositories), create a local copy of the Empower repositories (added as mirror repositories), or

use a product or fix image (created by the Software AG Installer or Software AG Update Manager and added as an image repository). Command Central will use the repositories to install the products and apply the fixes available for those products.

1. Choose a repository setup that works for you: [“ Understanding Software Stacks” on page 50](#)
2. Set up the repository credentials:
  - [“Create Credentials Aliases” on page 16](#)
  - "COMMON-CREDENTIALS Usage Notes" in [“Configuration Types for Command Central and Platform Manager OSGI ENGINE” on page 487](#) (the command line interface)
3. Add the repositories:
  - [“Connect to Software AG Repositories” on page 17](#); [“Create, Refresh, or Change Sources for a Mirror Repository of Products or Fixes” on page 17](#) (the web user interface)
  - [“sagcc add repository products” on page 390](#); [“sagcc add repository fixes” on page 386](#) (the command line interface)

## Step 2: Import the Product License Keys in Command Central

Import the license keys for the products that you want to install (or upgrade) and assign an alias to each license key: [“Import Product License Keys for Instances or Components” on page 19](#) and [“sagcc add license-tools keys” on page 325](#).

## Step 3: Configure the SSH Connection to Each Remote Machine

Check if the UNIX or Windows operating system on the remote machine has a Secure Shell (SSH) server running. Also, make sure that the system is configured for remote access with the user account for the Software AG products.

- Set up the connection to the remote machine and the credentials alias for the SSH user: [“Bootstrapping Platform Manager Using the Default SSH and WinRM Templates” on page 146](#); "COMMON-CREDENTIALS Usage Notes" in [“Configuration Types for Command Central and Platform Manager OSGI ENGINE” on page 487](#)
- Optional. Configure a substitute user: [“Bootstrapping on a Remote Machine with a Substitute User” on page 146](#)

## Step 4: Create the Stacks

Use one of the following options to create a stack:

- Create a stack from scratch. With this option, you create the stack and its layers using the default templates (from [sagdevops-templates](#)) or templates generated from existing installations that Command Central manages. For details about creating a stack from scratch, see [“Creating a Stack” on page 56](#), [“sagcc create stacks” on page 425](#), and [“sagcc create stacks layers” on page 428](#).

- Use auto-create stacks. With this option, Command Central automatically creates the stacks for the installations that you specify as described in [“Create Software Stacks from Existing Environments” on page 57](#) and [“sagcc create stacks auto” on page 426](#). Note that you can use this option only for standard product installations managed by Command Central.

### Step 5: Prepare the Property Files to Use with the Stacks

Create one or more properties files with the values that you must specify for the required input parameters of the layers in the stack. You can also customize other parameters in the layers with a properties file, for example if you want to specify different values for different environments. In the Stacks view of the Command Central web user interface, you can import the properties file(s) that you want to use for the layers in a stack. You can also export the parameters you specify for a layer into a properties file.

### Step 6: Run the Stacks

After you create a layer in a stack and specify the required properties for the layer, Command Central installs the products in the layer automatically on the specified host machines.

When you want to upgrade product installations through a stack, you can conduct a dry run of the upgrade to identify and troubleshoot issues before the actual upgrade. During the dry run, Command Central generates the migration templates for the upgrade, but does not apply them on the target machines. For instructions how to run the upgrade operation, see "Create an Upgraded Installation Using Command Central and Software Stacks (10.1 and Later)" in [Upgrading Software AG Products On Premises](#).

Note that if you have custom files and configurations that should get copied (such as adapter jar files), you must run the upgrade as follows:

1. Click **Dry run** to generate the migration templates.
2. Locate the migration templates in the Templates view and add the custom files or configurations in the generated templates.
3. Run the actual upgrade and click the option to re-use the generated templates.

### Step 7: Monitor and Troubleshoot the Template Application

Use the Jobs view in the Command Central web user interface to monitor the status of the template apply job and access the logs: [“Correcting a Failed Composite Template Apply Operation” on page 130](#).

## Update Licenses on a Number of Nodes with Templates

After completing the steps in this topic, you will have updated the licenses for the products in all installations managed by Command Central.

Before you begin, make sure that you have the license files for the product licenses that you want to update.

## Step 1 Import the new product license keys in Command Central

Import the new license keys for the products (and optional, assign an alias to each license key): “[Import Product License Keys for Instances or Components](#)” on page 19 and “[sagcc add license-tools keys](#)” on page 325.

## Step 2 Prepare a composite template to update the licenses

Choose one of the following options:

- Use a template with only a `templates: licenses:` section, see “licenses” in “[Templates](#)” on page 91. The template will copy the licenses (based on the alias defined for each license key in Command Central) to the correct location on the target file system.

This template is very simple but requires you to specify the exact target location to which to copy each license file.

**Tip:**

You can use the `sagcc get configuration data <node_alias> <component_ID> COMMON-LICLOC-<instanceID>` command to get the license location for a product instance.

- Use a template with:
  - A `templates: licenses:` section to copy the licenses (based on the aliases defined for the licenses in Command Central) on the remote target hosts
  - For each product instance, define a COMMON-LICLOC configuration instance to assign the license to the product. For the template syntax, see “products” in “[Templates](#)” on page 91 or the default templates in [sagdevops-templates](#).

This template is more complex, but does not require you to specify the target location to which to copy the product license files.

## Step 3 Apply the template

Apply the license update template using the “[sagcc exec templates composite apply](#)” on page 451 command.

# 5 Command Central GitHub Projects

---

- Automate Command Central Installation, Configuration, and Upgrade ..... 72
- Default Templates on GitHub ..... 72

## 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](https://www.github.com/SoftwareAG/sagdevops-cc-server).

### Default Templates on GitHub

---

On GitHub, you can find the following projects that will help you automate the installation, configuration, and upgrade of Software AG products through the Command Central default templates:

- [sagdevops-templates](#) - a collection of Command Central default micro templates for Software AG products
- [sagdevops-templates-repo](#) - a repository with assets (generated from the default micro templates) to import as layer definitions in Command Central

# 6 Using Micro Templates

---

■ What is a Micro Template? .....	74
■ When to Use Micro Templates and When a Composite Template? .....	74
■ Connecting to the Default Templates Repository .....	75
■ Designing Micro Templates .....	75
■ Using the Sample Micro Templates .....	76
■ Creating a User-Defined Micro Template .....	76
■ Exporting the Platform Manager of an Installation to a Template .....	77
■ Exporting an Installed Product Instance to a Micro Template .....	79
■ Adding Configurations in an Exported Micro Template .....	81
■ Exporting the Database for Products in an Installation to a Template .....	81
■ Validating a Micro Template .....	82

## 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).

## Connecting to the Default Templates Repository

You can connect to the default templates assets repository on GitHub to import the template assets into your local Command Central installation. You can also import template assets from a flatfile image of the default template assets repository. Note that the default template assets repository on GitHub [sagdevops-templates-repo](#) contains only a subset of the default templates included in the sagdevops-templates project. For details about the template assets available in sagdevops-templates-repo, see the project readme on GitHub.

To connect to the default template assets repository, go to the Repositories > Assets tab on the Command Central web user interface, click + and select "Connect to Software AG Templates Repository". In the **Type** field select one of the following:

- **Default templates on GitHub** - Connect to the default template assets repository on GitHub.
- **Local templates ZIP file** - Browse to the location on the file system of a zip archive file with the default template assets. For details about how to create the zip archive file, see the readme in the sagdevops-templates-repo project.

After you register the default template assets repository, Command Central imports the template assets from the repository into the local Command Central installation. You can view the list of imported templates and their template definitions in the Templates view of the Command Central web user interface.

Command Central also creates a layer type definition for all templates used in one of the layer type definitions in the [sag-cc-layer-defs](#) template. If a template you have imported is not used in sag-cc-layers-defs, you can locate the template in the imported templates list (in the Templates view) and click the **Layer Definitions** link in the information message to create a layer definition for the template. From the Stacks view, you can use the layer type definitions to create a run-time layer in an existing stack.

Command Central notifies you of changes in the default template assets repository on GitHub with an  icon next to the repository name on the Repositories > Assets page. If you click the  icon, Command Central will re-import the updated default template assets and overwrite the previously imported templates. If you edit a default template in your Command Central installation, save the template with a different alias and re-import it.

## 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.
    runtimes.skip.start: <rcId1>,<rcId2> # A list of IDs of the run-time components,
    which
                                # Command Central will not start when
applying the
                                # template. Use a comma (,) for list
separator.
                                # You can use a regular expression.
```

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](https://github.com/sagdevops-templates) project.

## 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](https://github.com/sagdevops-cc-server) project to automate setting up Command Central.

1. Define the micro template YAML definition. See [“Designing Micro Templates” on page 75](#).

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

2. Import the user-defined micro templates in Command Central from the **Templates** view in the Command Central web user interface or with the “[sagcc exec templates composite import](#)” on page 454 command.
3. Create a layer type definition for the imported micro template:
  - In the Command Central web user interface, go to **Layer Definitions** on the **Configuration** tab for the Command Central Server that hosts the imported template
  - Create an instance of the CCE-LAYER-TYPES configuration type that references the template with the “[sagcc create configuration data](#)” on page 233 command. For 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 238 to retrieve data for the same type of configuration instance you want to create. For example, run the following command to get the configuration data for the CCE-LAYER-TYPES-runtime-existing configuration instance.

```
sagcc get configuration data local OSGI-CCE-ENGINE RUNTIME-EXISTING --format
xml
```

You can also use a micro template to create the layer type configuration instances. See the sample layer definitions template at [sag-cc-layer-defs](#)

After importing 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 56.
- The CLI stacks and layers commands. See “[sagcc create stacks](#)” on page 425 and “[sagcc create stacks layers](#)” on page 428.

## Exporting the Platform Manager of an Installation to a Template

You can export the Platform Manager of an installation managed by Command Central to an infrastructure micro template. Command Central generates a template with the Platform Manager data, imports the generated template automatically, and creates a layer type definition for the template in the local Command Central installation. You can use the layer definition for the template to create an infrastructure layer in an existing stack.

In the Command Central web user interface, you access the **Export** wizard from the Overview page of an SPM instance by selecting the **Export infrastructure** option. By default, Command Central generates a template that includes the following data about the Platform Manager instance:

- Products, which includes the Platform Manager components and the product plugins installed on the SPM instance

- Fixes
- Configurations of the SPM instance

If you select the **Include Hostname** checkbox, the generated template will also include the hostname of the Platform Manager node, from which you export Platform Manager.

Note that the sections and parameters included in the generated template will depend on the release version of the Platform Manager installation from which you export the Platform Manager instance. For details about the template that Command Central generates based on the selected infrastructure data and the release version of the Platform Manager instance, see the "Usage Notes for Exporting a Platform Manager Instance" in [“sagcc exec templates composite generate” on page 457](#).

By default, if you choose to include configurations, Command Central exports all configuration instances for all configuration types from the selected Platform Manager instance and its child components. In the Select Configuration Types step of the wizard, you can select which configuration types for the Platform Manager instance and its child components to include in the generated template.

You can view the exported template in the Templates view of the Command Central web user interface. If you want to modify the template, you can download and edit it in a text editor, and then re-import the template in Command Central.

## Considerations when Exporting Platform Manager

When exporting a Platform Manager instance to an infrastructure template, you should consider the following:

- If you want to export a Platform Manager instance to a different host, but keep the same node alias name, select **Include Hostname** to include the hostname of the Platform Manager node in the exported template. You can use this approach when migrating a node.
- When exporting a Platform Manager instance with fixes, you must check the version of the Command Central bootstrapper that you will specify in the `cc.installer: ${}` parameter in the generated template. If you specify a bootstrapper that contains fixes with a higher version than the ones listed in the generated template, the fixes from the bootstrapper will get installed and will replace the ones from the template.
- You cannot apply an exported template with configurations on the same Platform Manager installation from which you exported the configurations. Software AG recommends using one virtual machine for the Platform Manager installation from which to export and another virtual machine for the Platform Manager installation in which to apply the exported template with configurations.
- If any of the configurations you export use full file paths, you must review the generated template and parameterize the configurations that include full file paths.

## Example

The `log4j.configurationFile` property in the following COMMON-SYSPROPS configuration uses a full installation directory path that gets exported as follows:

```
COMMON-JAVASYSPROPS:
COMMON-JAVASYSPROPS:
  osgi.install.area: "%OSGI_INSTALL_AREA%"
  log4j.configurationFile.stripquotes: 'TRUE'
  java.util.Arrays.useLegacyMergeSort: 'true'
  osgi.install.area.stripquotes: 'TRUE'
  log4j.configurationFile:
  "C:\SoftwareAG105\profiles\SPM\configuration\logging\log4j2.properties"
  java.io.tmpdir.stripquotes: 'TRUE'
```

You can replace the hardcoded installation directory by referencing the `nodes.install.dir` parameter, which is defined in the generated template:

```
log4j.configurationFile:
"${nodes.install.dir}/profiles/SPM/configuration/logging/log4j2.properties"
```

If you replace all hardcoded full file paths in the exported template with parameters, you can apply the template on another machine and in a different installation directory from the source installation directory. If you do not replace the full file paths in the exported template, you must apply the template on another machine and in the same installation directory as the source installation.

## Exporting an Installed Product Instance to a Micro Template

You can export a product instance, which is installed in an installation managed by Command Central, to a run-time micro template. Command Central generates a template with the product instance data, imports the generated template automatically, and creates a layer type definition for the template in the local Command Central installation. You can use the layer definition for the template to create a layer for the product instance in an existing stack.

In the Command Central web user interface, you access the **Export** wizard from the Overview page of a product instance by selecting the **Export runtime** option. By default, Command Central generates a template that includes the following data about the product instance:

- Instance properties
- Fixes
- Inline templates for the products hosted on the instance and their configurations
- Configurations of the product instance

If you do not want to include some of the product instance data, clear the checkbox for that data. For example, clear the **Include Instance** checkbox to indicate that the properties of the product instance should not get included in the generated template.

Note that the sections and parameters included in the generated template will depend on the release version of the installation from which you export the product instance. For details about

the template that Command Central generates based on the selected instance data and the release version of the product instance, see the "Usage Notes" in [“sagcc exec templates composite generate” on page 461](#).

By default, if you choose to include configurations, Command Central exports all configuration instances for all configuration types from the selected product instance and its child run-time components. In the **Select Configuration Types** step of the wizard, you can select which configuration types for the product instance and its child run-time components to include in the generated template.

You can view the exported template in the **Templates** view of the Command Central web user interface. If you want to modify the template, you can download and edit it in a text editor, and then re-import the template in Command Central.

## Considerations when Exporting Configurations

If you choose to export the configuration settings of a product instance to a template, you must consider the following:

- You cannot apply an exported template with configurations on the same Platform Manager installation from which you exported the configurations. Software AG recommends using one virtual machine for the Platform Manager installation with the product instance that you want to export and another virtual machine for the Platform Manager installation in which to apply the exported template with configurations.
- If any of the configurations use full file paths, you must review the generated template and parameterize the configurations that include full file paths.

### Example

The `log4j.configurationFile` property in the following COMMON-SYSPROPS configuration uses a full installation directory path that gets exported as follows:

```
COMMON-SYSPROPS:
COMMON-SYSPROPS:
  osgi.install.area: "%OSGI_INSTALL_AREA%"
  log4j.configurationFile.stripquotes: 'TRUE'
  osgi.install.area.stripquotes: 'TRUE'
  log4j.configurationFile:
  "/home/vmtest/sag/profiles/CTP/configuration/logging/log4j2.properties"
  java.io.tmpdir.stripquotes: 'TRUE'
```

You can replace the hardcoded installation directory with a parameter:

```
log4j.configurationFile:
  "${install.dir}/profiles/CTP/configuration/logging/log4j2.properties"
```

And add the `install.dir` parameter in the `environments: default:` section of the template definition:

```
environments:
  default:
    install.dir: /home/vmtest/sag
    ...
```

If you replace all hardcoded full file paths in the exported template with parameters, you can apply the template on another machine and in a different installation directory from the source installation directory. If you do not replace the full file paths in the exported template, you must apply the template on another machine and in the same installation directory as the source installation.

To watch a video that shows how to recreate an environment by exporting an installed product instance to a template and then creating a layer for this product instance in an existing stack, click <https://www.youtube.com/watch?v=RWkxWCgnbik&feature=youtu.be>.

## Adding Configurations in an Exported Micro Template

If you modify any of the product configurations after you have exported a product instance to a micro template, use the following steps to add the changed configuration in the exported micro template:

1. In the Command Central web user interface, go to **Environments > ALL > Instances** and click the instance, from which you want to export configuration data.
2. Go to the Configuration tab and select the required configuration type from the drop-down box.
3. Click the name of the required configuration in the table.
4. Click **Export** and copy the template snippet for the configuration.
5. Open the exported micro template in a text editor and navigate to the `products:productInstance:configuration:runtimeComponentId` section of the template. If a configuration with the same ID as the exported configuration:
  - Already exists, replace the existing configuration and paste the newly exported snippet.
  - Does not exist, paste the exported configuration snippet on a new line in `products:productInstance:configuration:runtimeComponentId`.
6. Save the template and re-import it into Command Central.

## Exporting the Database for Products in an Installation to a Template

You can export the database components and schemas configured for the products in an installation to a database micro template.

In the Command Central web user interface, you access the **Export** function from the Overview page of the **SPM** product instance in the installation by selecting the **Export database** option. Note that the **Export database** option is enabled only when Database Component Configurator is installed in an installation of release version 10.5 or higher. In the **Export Database to Template**

wizard, you must specify an alias name for the database template and the connection details of the database, such as database type, URL, and user credentials.

**Tip:**

To get the database connection details, go to a product instance (for example, Integration Server) and select **Configuration > Databases**. Click the name of an existing database configuration instance and copy the database connection details.

For the database types supported by Software AG products, see the *System Requirements for Software AG Products* document.

Command Central generates a template with the database components and schemas, and the connection details that you specified. Command Central imports the generated template automatically and creates a layer type definition for the database template in the local Command Central installation. You can use the layer definition for the template to create a database layer in an existing stack.

Note that the generated template does not include the database storage component. Before you apply the database template in an installation or use it to create a database layer in a product stack, you must ensure that the database storage component for the database is already created.

## 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. Command Central checks if the micro template referenced in the layer type definition complies with the definition of a micro template, as described in [“Designing Micro Templates” on page 75](#).
- After importing the template in Command Central, run the `“sagcc exec templates composite validate” on page 476` command with the alias of the micro template and check the validate command jobs in the Jobs view.

# 7 Using Composite Templates

---

■ About Automated Provisioning of Environments .....	84
■ What is a Composite Template? .....	84
■ Understanding the Composite Template Definition .....	85
■ Creating a Custom Composite Template .....	116
■ How Command Central Processes a Composite Template for Installations of Release 10.3 or Higher .....	116
■ How Command Central Processes a Composite Template for Installations of Release 10.1 or Lower .....	119
■ Importing a Template into Command Central .....	120
■ Creating a New Environment Using a Composite Template .....	120
■ Updating Command Central to the Latest Fix .....	121
■ Updating a Provisioned Environment Using the Same Composite Template .....	122
■ Migrating to a Different Host with Cloned Database .....	122
■ Migrating Using Disconnected Migration .....	124
■ Migrating a Product Instance .....	125
■ Migrating Platform Manager Using a Template .....	127
■ Monitoring the Status of a Composite Template .....	129
■ Validating a Composite Template .....	129
■ Correcting a Failed Composite Template Apply Operation .....	130
■ Composite Templates for 9.x Installations .....	131

## 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.

### 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 454](#) and [“sagcc exec templates composite apply” on page 451](#).

## 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: template1
description: |
  This is a basic template
  with a description.
dslVersion: "1.2"      # supported in Command Central 10.4 and higher
                  "1.1"      # supported in Command Central 10.3
                  "1.0"      # supported in Command Central 10.1 and lower
version: 0.1
```

Note that some of the instructions and parameters included in higher DSL versions are not supported by the lower versions of the DSL.

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.

## Options

### Template DSL "1.2"

With template DSL version "1.2", you can add the `options:` section, in which you specify parameters that modify how Command Central processes a template at the time of applying the template.

#### Syntax

```
alias: MyTemplate
dslVersion: "1.2"
options:
  abortOnConfigFailure: true | false  #OPTIONAL
  abortOnLayerStartupFailure: true | false  #OPTIONAL
  forceRestartWhenDone: true | false  #OPTIONAL
  startupLayers: true | false  #OPTIONAL
```

Based on the value of the `abortOnConfigFailure` parameter, Command Central determines whether to stop applying the template after the template fails to apply a product configuration:

- `abortOnLayerStartupFailure: true` Command Central fails the template application with ERROR status if a run-time component in a layer does not start. Command Central logs an error message in the `default.log`, which indicates the ID(s) and status of the run-time component(s) that failed to start.
- `abortOnLayerStartupFailure: false` (default) When starting a run-time component in a layer fails, the template reports a WARNING status. Command Central continues applying the template without interruption. The message in the Command Central `default.log` indicates the ID(s) and status of the run-time component(s) that failed to start.

If template application fails when `abortOnConfigFailure: true` and/or `abortOnLayerStartupFailure: true`, Command Central stops processing the current and all subsequent layers in the template.

See the “[Actions](#)” on page 111 section for details about how Command Central handles the execution of actions when a template fails with an ERROR status.

You can specify the `forceRestartWhenDone` parameter only when applying a template for product installation 10.3 or higher. Based on the value of this parameter, Command Central determines whether to restart all run-time components as the last step in applying the template:

- `forceRestartWhenDone: true` Command Central does not restart the run-time components when applying product configurations for each layer, but forces a restart of all run-time components at the end of the template application.
- `forceRestartWhenDone: false` (default) After applying product configurations for a layer, Command Central restarts only the run-time components in the layer that require a restart.

You can specify the `startupLayers` parameter only when applying a template for product installation 10.3 or higher. After applying a layer, Command Central determines whether to start all run-time components created for the layer as follows:

- `startupLayers: true` (default) Starts the run-time components.
- `startupLayers: false` Does not start the run-time components.

Note that you can specify the values of the parameters in the `options:` section by reference, so that you can provide the actual value at the time of applying the template (in a properties file or the command line interface):

```
options:
  abortOnConfigFailure: ${value}
  abortOnLayerStartupFailure: ${value}
  forceRestartWhenDone: ${value}
  startupLayers: ${value}
```

## Template DSL "1.1"

The template DSL versions "1.1" and "1.0" do not support the `options:` section and its parameters.

## 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 of 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.

When using a composite template for migration, you can add the migration registry clean-up parameters listed in the following example in an *environments/environment.type* section.

```
environments:  
  default:  
    env.type:  
      migration.pre.cleanup: true|false # Optional. When set to true, Command Central  
      cleans up                          # the migration registry before executing the  
      template.  
      migration.post.cleanup: true|false # Optional. When set to true, Command Central  
      cleans up                          # the migration registry after executing the  
      template.
```

When the parameters are not set or their value is `false`, Command Central automatically cleans up the migration registry 60 hours after a successful template execution. You can also specify these parameters as arguments of the `sagcc apply composite templates` command.

## 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 following table describes the how the parameter values in this example are resolved:

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
<i>environment.type</i> =bar	Error. The template does not include an <i>environment.type</i> with value "bar".

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

---

## 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
```

For each layer defined in the template, you can specify the credentials alias of the administrator user for the products in that layer. You can also specify whether the products require changing the administrator user password on the first login.

```
layers:
  <layerName>:
    productAdministrator:
      credentials: <administratorCredentialsAlias> # OPTIONAL The credentials
alias defined for the Administrator user credentials for the
# products in the layer. The
default is the ADMINISTRATOR credentials alias.
      changeOnFirstLogin: true|false # OPTIONAL Indicates whether the product
requires Administrator password change on the first login.
# Valid values: true (password change
required)| false (no password change required). Default: false
```

## 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:templateAlias:` sections, you can specify context parameters that resolve to values specific to the context in which you apply the template. However, if you define actions: under the `templates:` section and use context variables in the action file names, the context variables do not resolve automatically when you apply the template.

The following table lists the context parameters supported with DSL version "1.0" or higher:

Parameter	Description
<code>\${node.host}</code>	Resolves to the host of the node on which the template is provisioned.
<code>\${node.alias}</code>	Resolves to the alias of the node on which the template is provisioned.

Parameter	Description
<code>\${src.node.host}</code>	When <code>environment.mode=MIGRATION</code> , resolves to the host of the source node for the migration. When <code>environment.mode=PROVISION</code> , resolves to the same value as <code>node.host</code>
<code>\${migration.type}</code>	Resolves to the type of migration (overinstall, same host, or cross-host) required for the product instance.

The following table lists the context parameters supported with DSL version "1.1" or higher:

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>HOSTNAME</code> or <code>EXISTING</code> , the value of this parameter is empty.
<code>\${nodes[i]}</code>	Resolves to the node aliases of the infrastructure layer with an index number, for example <code>\${nodes[1]}</code>
<code>\${hosts[i]}</code>	Resolves to the hosts of the infrastructure layer with an index number, for example <code>\${hosts[1]}</code>
<code>\${nodes.count}</code>	Resolves to the number of nodes on which the infrastructure layer is applied. The value is 1 or higher, although 0 is also a valid value.  Note that you can also use this context parameter in the <code>layers:</code> section.
<code>\${node.index}</code>	Resolves to the index number of the node in the infrastructure layer. The value starts at 1 and goes as high as <code>\${nodes.count}</code> .
<code>\${myLayer.nodes[i]}</code>	Resolves to the node alias of <code>myLayer</code> with a node index number, for example <code>\${um.nodes[1]}</code> .
<code>\${myLayer.hosts[i]}</code>	Resolves to the node host of <code>myLayer</code> with a node index number, for example <code>\${um.hosts[1]}</code> .
<code>\${myLayer.nodes.count}</code>	Resolves to the number of nodes in <code>myLayer</code> . The value is 1 or higher, although 0 is also a valid value.  Note that you can also use this context parameter in the <code>layers:</code> section.
<code>\${myLayer.node.index}</code>	Resolves to 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> .

In the `templates:templateAlias:` sections, 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:
      configuration:
        runtimeComponentID:
          CONFIG-TYPE:
            CONFIG-TYPE-instance1:
              config_setting1: value
              config_setting2: value
              config_setting3: ${node.host}
              config_setting4: ${node.alias}
              config_setting5: ${src.node.host}
```

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:

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 the Command Central online help.

## fixes and support patches

Use the `fixes:` section to specify a list of fix names or Empower fix IDs for the fixes that you want to install. You can find the fix names, Empower fix IDs, or the exact version number of a fix using the [“sagcc list repository fixes content” on page 402](#) or when viewing the contents of a fix repository in the web user interface. If you specify only fix names, Command Central installs the latest available version of each fix and its dependencies from the fix repository defined in the template. For example:

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

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_10.3.0.0001-0001, wMFix.Product2Fix_10.3.0.0003-0123]
```

You can include the Empower fix ID in the following format `ProductCode_Version_Component_FixN`. For example:

```
fixes: [Product1Code_10.3_Core_Fix1, Product2Code_10.3_Fix3]
```

You can include a combination of fix names (with or without the exact fix version) or Empower fix IDs. For example:

```
fixes: [wMFix.Product2Fix, wMFix.Product1Fix_10.3.0.0001-0001, Product2Code_10.3_Fix3]
```

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 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). 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

## 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 for Installations of Release 10.3 or Higher” on page 116](#). However, when you want to use a composite template for migration, you must always include the [“Provision” on page 102](#) 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 is `false`.

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 and file directories 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 and file directories 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", "path/to/directory/to/exclude/*"]
    except: [list-of-file-names-excepted-from-patterns]
```

#### ■ rename

Renames the source installation directory to `migration_source_sourceVersion_installDir`, where `sourceVersion` is the version of the source installation and `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_sourceVersion_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}
```

## Specifying Product Migration Settings In a Properties File

You can also include the migration settings in a properties file, 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: HOSTNAME|EXISTING`
- An indexed node alias when `policy: INDEX`

For details about `aliasMapping:policy:` see [“Nodes” on page 105](#).

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 HOSTNAME 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: HOSTNAME

```

## Maintenance

Optional. When applying a template in maintenance mode, by default Command Central pauses and stops the run-time components in the installation before applying the template. Include a maintenance section in the template when you want to change this default behavior.

```

alias: my-alias
description: Update products to the latest fix
maintenance:
  options:
    pause: true | false      # "true" (default) - pause all runtimes, before stopping
                             them
                             # "false" - do not pause the runtimes

```

```

shutdown: true | false # "true" (default)- stop all runtimes before installing
fixes
                                # "false" - do not stop the runtimes
resume: true | false # "true" (default) - execute the resume lifecycle operation
on all runtimes
                                # "false" - do not execute the resume lifecycle operation
on runtimes

```

When applying a template with the `resume` option set to `true`, if a runtime does not support the resume lifecycle operation, or is already online, Command Central skips the runtime when executing the resume operation.

## 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: HOSTNAME|EXISTING`  
`HOSTNAME` (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`. Use the `delimiter:` parameter if you want to specify a custom delimiter for the generated indexed node alias. For example:

```
nodes:
  default:
    aliasMapping:
      policy: INDEX
      prefix: node
      delimiter: "_"
```

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 102](#) section.

### **credentials: \${spm.credentials.alias}**

In the `nodes:default:default:` section, you must specify the credentials alias for the Platform Manager administrator user credentials. Beginning with Platform Manager release version 10.11, the `credentials` parameter is required and by default uses the `ADMINISTRATOR` 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 define a credentials alias for the custom administrator user credentials. For example, 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 487](#).

When bootstrapping nodes with version:

- 10.11 and higher, you should use the `ADMINISTRATOR` credentials alias or a custom credentials alias with a strong password. Note that the `ADMINISTRATOR` credentials alias does not have a default password value and you must specify a strong administrator password before using the `ADMINISTRATOR` credentials. For more information, see [“Default and Custom Product Administrator User Passwords” on page 31](#).
- 10.7 and lower, by default the template uses the `DEFAULT_ADMINISTRATOR` credentials alias, which includes the default password. Software AG strongly recommends that you replace the default password with a strong administrator password.

## 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 488](#).

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

  envType1:
    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.

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.
  substituteUserCredentials: ${subst.credentials.alias} # For remote bootstrap: the
                                                         # alias for the substitute
user                                     # credentials to use to
install                                 # and start 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.
```

For more information about how to use `substituteUserCredentials:` and `credentials:`, see [“Bootstrapping on a Remote Machine with a Substitute User” on page 146](#).

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. To list the available bootstrap installers, run the following CLI command: `sagcc list provisioning bootstrap installers`.

## Usage Notes

- When you use the Command Central bootstrap installer for the nodes bootstrap, only Platform Manager is installed on the nodes, without the CLI. To install the CLI with the same template, you must include an inline template that installs the CLI as a product.
- 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:**

By default, the Command Central bootstrapper installs Java on the remote host and the SSH user connects to the Java installation. If required, you can use an existing Java installation on the remote host by adding a `javaPath` property in the `bootstrapInfo` section to specify the Java location as follows: `javaPath: /path/to/java`

- You can define the `${install.dir}` parameter as a variable in the `environments` section that points to `${user.home}`, for example:

```
environments:
  default:
    cc.installer: cc-def-10.7-milestone-w64.bat
    install.dir: ${user.home}\sag102
  ...
nodes:
  default:
    default:
      port: ${spm.port}
      secure: false
      bootstrapInfo:
        installDir: ${install.dir}    # Installation directory
        installer: ${cc.installer}
  ...
```

To resolve `${user.home}` correctly in Windows, you must start the Command Central and Platform Manager Windows services from a local user account with administrator permissions, not from a "Local System" user account.

## 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 488](#).

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 487](#).

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 examples of repository definitions, see the [sag-cc-repos](#), [sag-cc-mirrors](#), and [sag-cc-repos-assets](#) sample templates .

## 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 in a different way. The following table describes how the actions in the different sections are executed:

The actions in this section	are executed
<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 following table lists and describes the parameters, included in the `actions` section:

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> :

Parameter	Description
<code>description:</code>	Optional. States the goal of the action.
<code>phase: {pre   post}</code>	<p>Optional. 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. The actions defined in the default sub-sections of top level sections are executed as follows:</p> <ul style="list-style-type: none"> <li>■ Default actions with the <code>phase=pre</code> parameters are executed before regular <code>pre</code> actions.</li> <li>■ Default actions with the <code>phase=post</code> parameters are executed after regular <code>post</code> actions.</li> </ul> <p>Default: <code>phase: post</code></p>
<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"> <li>■ <code>true</code> (default) - The composite template apply operation fails.</li> <li>■ <code>false</code> - The composite template apply operation ignores the error and continues processing the composite template.</li> </ul>
<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> <p>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.</p>
<code>file:</code>	<p>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></p>

Parameter	Description
<code>target: POSIX   WINDOWS</code>	<p>Optional. Indicates on which operating systems to execute the action. Valid values:</p> <ul style="list-style-type: none"><li>■ <code>POSIX</code> (default) - the action is executed on POSIX-compliant operating systems.</li><li>■ <code>WINDOWS</code> - the action is executed only on Windows operating systems. This value is not supported for remote actions.</li></ul>
<code>namePrefix: hexTimestamp   none</code>	<p>Optional. Indicates whether Command Central adds a prefix to the names of action files created on the file system. Valid values:</p> <ul style="list-style-type: none"><li>■ <code>hexTimestamp</code> (default) - the filename for an action is prefixed by a hexadecimal representation of the timestamp at the time of template evaluation.</li><li>■ <code>none</code> - the filename for an action does not get a prefix.</li></ul>
<code>skipOnTemplateError: {true   false}</code>	<p>Optional. Indicates whether Command Central executes the action if applying the template fails with an <code>ERROR</code> status. Valid values:</p> <ul style="list-style-type: none"><li>■ <code>true</code> - does not execute the action.</li><li>■ <code>false</code> (default) - executes the action before stopping the template application.</li></ul>
<code>verbose</code>	<p>Optional. Indicates whether to include the standard output from the shell actions defined in the template in the Command Central logs. Valid values:</p> <ul style="list-style-type: none"><li>■ <code>true</code> - include the standard output.</li><li>■ <code>false</code> (default) - do not include the standard output.</li></ul>

## 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>.
- If applying the template fails at a certain operation, all actions pending execution after that operation are discarded and will not be executed. For example, if the template fails at the provisioning stage and the template includes an environment action with `phase: POST`, the action will not get executed.
- By default Command Central completes all post actions defined in the template even when Command Central stops the template application because of a failed configuration or a layer startup failure. When you want to skip executing an action if applying the template fails, set `skipOnTemplateError: true`
- Note that when setting `verbose: true` for actions that contain sensitive information (such as plain text passwords), the sensitive information will get included in the Command Central logs.

## 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`” command 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. The standard output from the “`cleanUpEnvDefault`” action will get included in the Command Central logs.

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

## 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 your through the steps required to develop a basic template.

You can also use the sample templates from the <https://github.com/SoftwareAG/sagdevops-templates> project on github as a source template or example for creating custom templates. Read and follow the instructions in the readme file and wiki of the project and then select and modify the sample template that matches or is close to your requirements. Use the sample templates from <https://github.com/SoftwareAG/sagdevops-templates> to set up environments with Software AG products of release 10.1 and higher.

**Note:**

Use the sample templates in [sagdevops-9.x-template-samples](#) to set up basic or typical product environments with Software AG products of release 9.x.

## How Command Central Processes a Composite Template for Installations of Release 10.3 or Higher

---

Command Central applies a composite template based on the processing mode specified in the `environment.mode= {provision | migration | maintenance}` 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.

### `environment.mode=provision|migration`

When applying a template in `provision` or `migration` mode, Command Central executes the following steps. Some steps are only executed in `migration` mode.

#### Step 1

Validates the composite template, as described in the [“Validating a Composite Template” on page 129](#) 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.

- a. Create a template with products, fixes, configuration, and files from each source node.
- b. PAUSE all run-time components:
  - 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.
- c. Shut down all run-time components, including Platform Manager.
 

Note that 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, this option is set to `true`.
- d. Rename the source installation directory to `migration_source_sourceVersion_installDir` when migrating on the same host and installation directory.

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. Command Central identifies the nodes, on which to provision or migrate a layer, from the `provision/environment.type` section.

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. Command Central migrates each product instance using the migration utility of the product. For details about the product instance properties that you can migrate with a template and how they map to the migration settings of the product migration utilities, see "Migrating a Product Instance".

After creating or migrating a product instance, Command Central updates the product instance with the required fixes.

## Step 7

Processes files that are defined inline and applies configuration settings.

To configure the products, Command Central goes through the following sub-processing steps for each layer in turn:

1. Apply the configuration properties defined for each run-time component.
2. If a configuration property requires a restart of the run-time component, the component reports `PENDING_RESTART` status.

3. Platform Manager restarts only the run-time components that report PENDING\_RESTART.
4. Start all run-time components created from all inline templates in a layer and wait for the components to start running with ONLINE status.

Command Central repeats the same steps for the next layer defined in the template. For example, if the layers section defines `um` and `is` layers, Command Central applies the configuration properties for the products in the `um` layer and waits for all run-time components in the `um` layer to start, before moving on to apply configuration on the run-time components in the `is` layer.

## environment.mode=maintenance

When applying a template in maintenance mode, Command Central runs only the following default steps to update installed products to their current fixes:

### Step 1

Validates the composite template, as described in the [“Validating a Composite Template” on page 129](#) topic.

### Step 2

Pauses and then stops the run-time components on all nodes included in the `provision:` section of the template. Installs the fixes for the products (and product plug-ins) defined in the inline templates.

### Step 3

Starts the run-time components on all nodes. You can control this behavior by adding an `options:` section in the template and setting the `startupLayers:` parameter, as described in [“Options” on page 86](#).

### Step 4

Executes the resume lifecycle operation on all runtimes.

To control the behavior of the template in maintenance mode, include the `maintenance:` section in the template definition, as described in [“Maintenance” on page 104](#).

## How Command Central Processes a Composite Template for Installations of Release 10.1 or Lower

---

Command Central goes through the same tasks when processing templates for installations of release 10.1 or lower, except for Step 7. In the product configuration step, Command Central applies configurations on all run-time components in all layers. At the end of the template application, Command Central forces *all* run-time components in the installation to restart.

## Importing a Template into Command Central

---

Before you can apply a template or use it to create a layer in a stack installation, you must import the template into Command Central. You can import a composite or micro template either from the Templates view in the Command Central web user interface or using the “[sagcc exec template composite import](#)” on page 454 CLI command. All templates imported into Command Central are listed in the Templates view in the Command Central web user interface.

After importing a template, you can:

- For a micro template, create a layer type definition for the template (from **Layer Definitions** on the **Configuration** tab for the Command Central Server that hosts the imported template) and then create a run-time layer that references this template.
- Click the alias of a template on the Templates page and read or copy the template definition. Note that you cannot edit the template definition in the text field.
- Use the **Copy** button to copy just the template alias, for example to use the alias in a template CLI command.

Note that the description in the **Description** column is automatically extracted from the `description:` instruction in the template definition. If you want Command Central to display a description for a template in the web user interface, you must add the description in the template definition.

To modify a template that you have imported into Command Central, you should first download the template as a ZIP archive (the template definition with all supporting files) or a YAML file (just the template definition). After you make the required changes, re-import the template and confirm that you want to overwrite the existing template with the same alias.

You can also download an imported template as an asset archive (that includes the template ZIP archive and an ACDL file).

## 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 **Templates** view in the Command Central web user interface or the [“sagcc exec templates composite import” on page 454](#) command. For example:

```
sagcc exec templates composite import -i template.yaml
```

4. Optional. Verify that your template is now registered in Command Central and listed in the **Templates** view. You can also verify with the [“sagcc list templates composite” on page 479](#) command, for example:

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

5. Apply the template by executing the [“sagcc exec templates composite apply” on page 451](#) 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 default.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 129](#).

## Updating Command Central to the Latest Fix

Software AG does not recommend using a composite template to update Command Central to the latest fix. You should use one of the following methods instead:

- Download the latest version of the Command Central bootstrap installer and run the bootstrapper by using the `-d path` argument to point to the existing installation directory. The latest version of the Command Central bootstrap installer includes the latest released fix. For instructions about downloading and running the Command Central bootstrap installer, see [“Install Command Central” on page 13](#).
- Run the [“sagcc exec provisioning fixes install” on page 364](#) CLI command and in the `repo_name` argument specify the name of a user-defined fix mirror or image repository, in which you have included the Command Central fix that you want to install. For example, you can locate the latest Command Central fix on Empower > Knowledge Center, create a fix image with Software AG Update Manager, and then add the image as a fix mirror or image repository in Command Central.

### Important:

Note that fix mirror repositories created against the Empower Software Download Center (SDC) product repository do not contain Command Central fixes, because Command Central is not available as a product in the SDC repository (that is you can only download the Command Central bootstrapper from the SDC, but not Command Central as a product). You must always create and specify a user-defined fix mirror or image repository when installing fixes to update Command Central.

## 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` on page 450 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 **Templates** view in the Command Central web user interface or the `sagcc exec templates composite import` on page 454 command.
5. Apply the updated composite template with the `sagcc exec templates composite apply` on page 451 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 105.

## 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 116.

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}
```

5. 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}
```

6. 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:

- a. On the Command Central host, in `CommandCentral_directory/profiles/CCE/data/migration/source/migration_source_templateAlias_nodeAlias`
- b. 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
```

7. 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.

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 116](#).
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 `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.

## Migrating a Product Instance

When migrating a product instance, you can define the instance properties you want to migrate in a template. The instance properties to include in the template depend on the product that you migrate and how you migrate it, for example as a single product instance or in a cluster. For details about the migration settings required by the migration utility of each product, see *Upgrading Software AG Products On Premises*.

```
environments:
  default:
    # Required. Name of the new instance on the target node
    # (and name of the source instance, when "src.instance.name:" is not included in
    the template).
    instance.name: newInstance
    # Name of the instance to migrate.
    src.instance.name: oldInstance

    # Only for MWS or UM instances in a cluster. The name of the new cluster node that
    hosts the instance.
    node.alias: newNodeName

    # A new host name if required, for example if you want to rename the old host.
    node.host: newHostName

    # Only for MWS or UM instances in a cluster. The name of the old cluster node.
    src.node.alias: oldNodeName

    # The name of the old host if required, for example if you want to rename the old
    host.
    src.node.host: oldHostName

    # The version of the source and target nodes if required.
    src.version: sourceNodeVersion
    dest.version: targetNodeVersion

    # Required. The JDBC URL and user credentials of the cloned database.
```

```

db.url: {}
db.username: {}
db.password: {}
...
templates:
  templateAlias
  products:
    productID:
      ${instance.name}:
        src.instance.name: ${src.instance.name}
        node.alias: ${node.alias}
        node.host: ${node.host}
        src.node.alias: ${src.node.alias}
        src.node.host: ${src.node.host}
        src.version: ${src.version}
        dest.version: ${dest.version}
        db.url: "${db.url}"
        db.username: ${db.username}
        db.password: ${db.password}

```

The following table shows how Command Central maps the instance properties from the template to the migration settings of the product utilities when migrating an instance:

Instance property in template	Maps to this migration utility setting
instanceName	-newInstanceName or if src.instance.name is not included, maps to -instanceName
src.instance.name	-instanceName
node.alias	-newNodeName
node.host	-newHostName
src.node.alias	-oldNodeName
src.node.host	-oldHostName
src.version	-srcVersion
dest.version	-destVersion
db.url	-cloneDbUrl
db.username	-cloneDbUser
db.password	-cloneDbPassword

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

## Changing an Instance Name During Migration

When migrating a product instance you can control whether the instance has the same name in the target installation as the name it had in the source installation. If you want to keep the name

of a product instance, define only an `instance.name`: instruction in the template that you intend to use for migrating the product installation. For example, to keep the name of the source instance "is1":

```
templates:
  is-server:
    products:
      integrationServer:
        is1:      # The name of the source instance that gets migrated and
                  # the new product instance in the target installation have the same
                  name, "is1".
```

If you want to change the name of a product instance in the target installation, define `instance.name`: and `src.instance.name`: in the template. The `instance.name` is used as the name of the new instance and the value of `src.instance.name` is the name of the old instance. For example, to change the name of the source instance from "is1" to "is2":

```
templates:
  is-server:
    products:
      integrationServer:
        is2:      # The name of the new product instance
                  # in the target installation will be "is2".
        src.instance.name: is1 # The name of the source instance that gets migrated
is "is1".
```

## Migrating Platform Manager Using a Template

You can migrate Platform Manager version 10.7 and higher through a composite template. For example, you can migrate the Platform Manager configurations (such as the proxy server configuration, SSL certificates, and user management settings) from a source installation into a target installation of the same or higher release version. For more information about the Platform Manager configurations and data that you can migrate, see [“Upgrade a Standalone Product Installation” on page 46](#).

Before you migrate the Platform Manager configurations from the source to the target installation, make sure that:

- If Platform Manager is installed on the target machine, it has not been started yet.
- The Command Central bootstrap installer for the target Platform Manager version is added in the target Command Central.
- If using a source archive to migrate the Platform Manager configurations to a different host, copy the ZIP source file to the target Platform Manager machine.

To migrate the Platform Manager configurations from the source installation to the target installation:

1. In the composite template, set the Platform Manager bootstrap parameters in the `nodes:default:default:bootstrapInfo:` section as follows:

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

```

    port: ${nodes.spm.port}           # The port of the source Platform
    Manager.
    bootstrapInfo:
      installer: ${cc.installer}      # The file name of the Command
    Central bootstrap installer to use for
                                     # the target Platform Manager
    version.
      installDir: ${install_dir}     # The installation directory of
    the target Platform Manager.
      port: ${nodes.ssh.port}
      credentials: spm.SSH.credentials

```

**Important:**

Do not change the Platform Manager port when migrating Platform Manager configurations, because the source and target Platform Managers must have the same port number and type.

2. Import the composite template in Command Central.
3. Apply the composite template in migration mode by running the apply composite templates command with the `environment.mode=migration` argument.

The Command Central bootstrap installer (specified in the `installer:` parameter in the template) will run the Platform Manager migration utility, which migrates all the configurations from the source Platform Manager installation to the target Platform Manager installation.

**Example**

To use the template with alias "spm-migration" to migrate the Platform Manager configurations from a 10.5 source installation to the target Platform Manager with version 10.7, located in the "C:\sag107\cc" directory:

1. Define the following bootstrap parameters in the "spm-migration" template:

```

...
nodes:
  default:
    default:
      ...
      bootstrapInfo:
        installer: cc-def-10.7-fix1-w64.exe # The file name of the bootstrap
    installer to use for the
                                     # target Platform Manager version
    10.7.
        installDir: C:\sag107\cc          # The installation directory of the
    target Platform Manager.
        credentials: defaultSSHcredentials
        port: 22

```

2. Import the "spm-migration" template in Command Central and run the following command to apply the template in migration mode:

```

sagcc exec templates composite apply spm-migration
environment.mode=migration spm.port=8093 src.install.dir=C:\sag105\cc
install.dir=C:\sag107\cc -s rubicon

```

When applying a composite template in migration mode, if the `installer:` parameter points to a bootstrap installer with version 10.5 or lower, or the `installDir:` parameter points to a target Platform Manager installation with version 10.5 or lower, Command Central does not migrate the Platform Manager configurations. You can only migrate Platform Manager 10.5 or lower by running the Command Central bootstrap installer 10.7 or higher at the command line.

## 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 or by running the [“`sagcc list jobmanager jobs`” on page 303](#) command. If a sub-job of the main composite template apply job fails, 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.

## 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 476](#).

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 517](#).

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.
- If the template installs products with version 10.11 or higher, validates that:
  - The ADMINISTRATOR or custom credentials alias defined in the template for the product administrator user has a password value.
  - The `nodes:default:default:credentials:` parameter uses the ADMINISTRATOR credentials alias or a custom credentials alias.
- 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.
- If the DSL version of the template is 1.0, checks if the alias mapping policy defined in `nodes:default:aliasMapping:policy:` is `HOSTNAME`.
- Validates that the node aliases are resolved successfully depending on the alias mapping policy defined in `nodes:default:aliasMapping:policy:`.
- Validates that the node aliases are resolved successfully depending on the alias mapping policy defined in `nodes:default:aliasMapping:policy:`.
- Validates the Platform Manager node as follows:
  - If the template does not include a `bootstrapInfo:` section for the node, checks if the node already exists in the Command Central landscape.
  - If the template includes a `bootstrapInfo:` section for the node, verifies that the node does not exist in the Command Central landscape and the alias of the infrastructure layer does not contain unresolved variables.
  - If the mapping policy defined for the node in `nodes:default:aliasMapping:policy:` is `EXISTING`, verifies that the node already exists in the Command Central landscape.
- Validates that the bootstrap installer defined in the `installer:` instruction is registered in Command Central.
- If an inline template contains the `patches:` section, checks that the DSL version of the template is 1.1 or higher.
- If the template contains the `options:` section, checks that the DSL version of the template is set to 1.2.
- If a remote action is defined in the template, checks that the SSH credentials for the remote node are also defined in the template.
- Writes the validation report to the `default.log`.

## 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.

## Composite Templates for 9.x Installations

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.

Use the sample composite templates in <https://github.com/SoftwareAG/sagdevops-9.x-template-samples> only to set up basic or typical product environments with Software AG products of release 9.x. The following table lists the sample 9.x 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"> <li>■ dev</li> <li>■ server</li> <li>■ cluster</li> </ul>	<ul style="list-style-type: none"> <li>■ database connection</li> <li>■ host names</li> <li>■ database server administrator user credentials</li> <li>■ license key aliases for Universal Messaging, Terracotta Server Array, Integration Server, Rules Engine</li> </ul>	<p>Provisions a typical BPMS environment, configured with the following layers:</p> <ul style="list-style-type: none"> <li>■ messaging (Universal Messaging)</li> <li>■ cache (Terracotta Server Array)</li> <li>■ bpm (Integration Server/Process Engine/Rules Engine/Monitor)</li> <li>■ presentation (My webMethods Server/Task Engine/Rules Engine/Monitor)</li> </ul> <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"> <li>■ default</li> <li>■ dev</li> </ul>	<ul style="list-style-type: none"> <li>■ database connection</li> <li>■ database server administrator user credentials</li> </ul>	<p>Installs the Database Component Configurator (DBC) on a remote installation and uses DBC to create a new database on a supported database server.</p>

Template alias	Environment Types	Required Parameters	Description
		<ul style="list-style-type: none"> <li>■ database components</li> <li>■ database products</li> </ul>	<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>
is-layer	<ul style="list-style-type: none"> <li>■ dev</li> <li>■ server</li> <li>■ layer</li> <li>■ cluster</li> </ul>	<ul style="list-style-type: none"> <li>■ database connection</li> <li>■ host names</li> <li>■ Terracotta Server Array URL</li> <li>■ database server administrator user credentials</li> <li>■ license key alias for Integration Server</li> <li>■ license key alias for Terracotta</li> </ul>	<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"> <li>■ dev</li> <li>■ server</li> <li>■ cluster</li> </ul>	<ul style="list-style-type: none"> <li>■ database connection</li> <li>■ host names</li> <li>■ database server administrator user credentials</li> </ul>	<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 bootstrapping installations or</p>

Template alias	Environment Types	Required Parameters	Description
			executing template-provisioning operations.
			Apply this template first on the master Command Central installation that has Internet access to the Empower Product Support Website.
spm-layer	<ul style="list-style-type: none"> <li>■ default</li> <li>■ dev</li> <li>■ server</li> <li>■ layer</li> </ul>	<ul style="list-style-type: none"> <li>■ SSH connection</li> <li>■ host names</li> </ul>	<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"> <li>■ dev</li> <li>■ server</li> <li>■ cluster</li> </ul>	<ul style="list-style-type: none"> <li>■ host names</li> <li>■ license key alias for Terracotta Server</li> </ul>	Provisions a standalone local or remote Terracotta Server or a two-server master/slave cluster on remote machines.
um-layer	<ul style="list-style-type: none"> <li>■ dev</li> <li>■ server</li> <li>■ cluster</li> </ul>	<ul style="list-style-type: none"> <li>■ host names</li> <li>■ license key alias for Universal Messaging</li> </ul>	<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>

## Database Actions in the 9.x Template Samples

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 Template](#)” on page 152. 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>.

The following table lists the aliases of the templates that create or migrate database components or schemas and the parameters that are used in each template.

Template alias	Parameters	Database component or schema created or migrated
bpms	components: [STR]	Storage and all components supported by the Database Component Configurator.
	products: [IS,BPM,MWS]	Database schemas for Integration Server, My webMethods Server, and Process Engine.
is-layer	components: [STR]	Storage and all components supported by the Database Component Configurator.
	products: [IS]	Database schema for Integration Server.
mws-layer	components: [STR]	Storage and all components supported by the Database Component Configurator.
	products: [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 [“Composite Templates for 9.x Installations” on page 131](#).

## 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 325 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 on the TECHcommunity website: [Using Cygwin to Configure OpenSSH When Installing Platform Manager on a Remote Windows Machine](#).

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 135.

## 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 136](#) 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 116](#).

## Understanding the is-layer Template Definition

### Environments

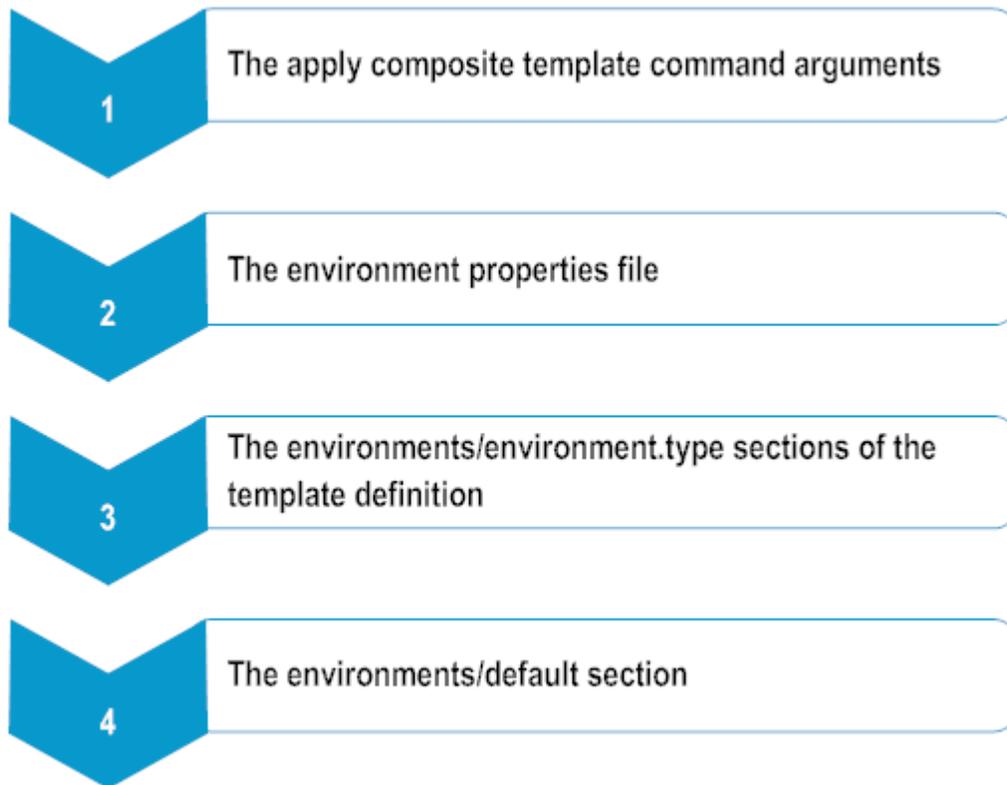
In this section you specify what type of environments you want to create on the target hosts. The is-layer template defines the following types of environments:

- `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. The composite template applies the apply composite template command arguments first, the

environment properties file second, the environment.type sections of the template definition third, and the environments/default section fourth.



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.
2. Identifies the Integration Server product ID from the `products` section and installs Integration Server.

3. 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.
4. 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.
5. 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 130.



## 8 Bootstrapping Platform Manager on a Remote Machine Using a Template

---

- [Bootstrapping Platform Manager Using the Default SSH and WinRM Templates .....](#) 146
- [Bootstrapping on a Remote Machine with a Substitute User .....](#) 146

## Bootstrapping Platform Manager Using the Default SSH and WinRM Templates

---

The method you choose to bootstrap Platform Manager on a remote machine depends on whether the remote machine has an SSH server. With Command Central 10.5 and higher, you can bootstrap Platform Manager on a remote machine using:

- The [sag-spm-boot-ssh](#) infrastructure micro template. Read the template readme and ensure that the Command Central host machine and the remote machine(s) meet the requirements described in the readme.
- A custom infrastructure micro template, generated by exporting an existing Platform Manager instance. For details about generating a custom infrastructure template see, ["Exporting the Platform Manager of an Installation to a Template"](#) on page 77.

Before you apply an infrastructure micro template that bootstraps Platform Manager, you must:

- Define a credentials alias for the SSH user account that you use to connect to the remote machine, for example by using the [sag-cc-creds-dev](#). Note that Command Central supports only the PKCS #1 definition of the ASN.1 syntax for representing the SSH private keys.
- When using a generated infrastructure micro template, ensure that the target machine is compatible with the source machine, from which you exported the template. The target machine should run on the same operating system as the source machine.
- When using the "sag-spm-boot-ssh" template, read the template readme and ensure that the Command Central host machine and the remote machine(s) meet the requirements described in the readme.
- Ensure that the UNIX or Windows operating system on the target machine has a Secure Shell (SSH) server running and the system is configured for remote access with the user account for the Software AG products. You must use Cygwin to configure OpenSSH on Windows operating systems. For information about how to install Cygwin, go to <https://cygwin.com/install.html> or follow the instructions in the [Using Cygwin to Configure OpenSSH When Installing Platform Manager on a Remote Windows Machine](#) article located on the TECHcommunity.

### **Important:**

The instructions in the article also apply when you want to use a composite template to bootstrap Platform Manager on a remote Windows machine.

If the target machine does not have an SSH server, you must download the Command Central bootstrapper to the target machine and install only Platform Manager on the remote machine as described in ["About Installing Platform Manager"](#) on page 36.

## Bootstrapping on a Remote Machine with a Substitute User

---

You can include a substitute user for bootstrapping operations on a remote machine. When including a substitute user, you have two users that perform different remote operations on the target machine:

- A user that connects over SSH to the remote machine and copies the bootstrapper on the machine.
- A substitute user that installs and starts Platform Manager.

In this case, add the credentials for both users in the `nodes/default/default/bootstrapInfo` section of the infrastructure micro template or composite template that you use to bootstrap Platform Manager on a remote machine.

Before you add a substitute user in the template, consider the following requirements:

- The remote machine must have a POSIX-compliant operating system. This function is not supported on Windows operating systems.
- The substitute user must have read permissions for the home directory of the SSH user.
- The SSH user must have read permissions for the home directory of the substitute user. You can use the template with the substitute user to bootstrap new Platform Manager installations or migrate existing ones.

To define a substitute user, add the `substituteUserCredentials:` section in the `nodes/default/default/bootstrapInfo` section of the template. You can add the substitute user credentials as `userName:` and `password:` parameters or map the substitute user to a credentials alias, for example `substituteUserCredentials: ${subst.credentials.alias}`. The following example shows a sample template snippet with a `substituteUserCredentials:` section that references a credentials alias parameter:

```
environments:
  default:
    install.dir: /opt/softwareag
    bootstrapper: cc-def-10.5-milestone-lnxamd64.sh
    subst.credentials.alias: ${substitute-user}
    os.credentials.alias: ${ssh-user}
  ...
nodes:
  default:
    default:
      port: 8093 # Default Platform Manager HTTPS port.
      secure: true # Indicates that the connection will be over SSL.
      bootstrapInfo:
        installDir: ${install.dir} # The installation directory.
        installer: ${bootstrapper} # Name of the Command Central bootstrapper to use.
        substituteUserCredentials: ${subst.credentials.alias} # Substitute user
credentials.
      credentials: ${os.credentials.alias} # SSH connection credentials.
  ...
```



# 9 Encrypting Passwords in Templates

---

■	Encrypting Passwords in Templates .....	150
---	---	-----

## Encrypting Passwords in Templates

---

You can secure passwords that you use in templates by encrypting the password with the “[sagcc exec security encrypt](#)” on [page 420](#) command and using the encrypted value in the template or properties file. Note that you must enclose the encrypted password value in quotes when you add it in a template YAML file.

If you have an existing product configuration with a password, in the Command Central web user interface, you can navigate to the required product configuration and click Export. The password value in the exported template snippet is encrypted. If you encrypt a password using this method, Software AG recommends to use the encrypted value only with the exported template snippet.

If you apply a template with encrypted passwords to provision installations with version:

- 10.5 and higher, the password is decrypted on the Platform Manager side.
- 10.4 and lower, the password is decrypted on the Command Central side.

# 10 Managing Database Components Using a Template

---

■ Managing Database Components Using a Template .....	152
■ Supported Database Component Configurator Parameters .....	156

## Managing Database Components Using a Template

You can manage the database components (storage, user, and schema) for Software AG products using a Command Central template as follows:

- Use a dedicated database template to create a database layer. This approach is recommended when creating a product stack in Command Central. The [sagdevops-templates](#) project on Github has sample database templates for several supported databases.
- Define a database layer with database actions in the `layers:` section of a composite or run-time micro template. This approach ensures that when you install the database script of a product, the actual product is also installed in the same product installation.

The database layer typically includes one or more database action sections with user-defined action names. Each section can contain the parameters that are listed in the mapping table in “[Supported Database Component Configuration Parameters](#)” on page 156 and required to manage the database storage, user, and schema.

### Example Structure of a Template with a Database Layer and a Run-time Layer

```
environments:
  default:
    is.integrationserver.license.key: "*_PIE_10.*_*_*"
    is.instance.name: default
    is.primary.port: 5555
    is.diagnostic.port: 9999
    is.jmx.port: 8094
    is.fixes:
      - wMFix.IntegrationServer.Core
    db.host: ${}
    db.version: 10.5 # Specifies the release version of Integration Server.
    db.type: mysql
    db.port: 3306
    db.admin.username: ${}
    db.admin.password: ${}
    db.admin.url: "jdbc:mysql://${db.host}:${db.port}"
    db.url: "jdbc:mysql://${db.host}:${db.port}/${db.name}"
    db.name: webm
    db.username: ${}
    db.password: ${}
    db.str.components: # List the ID of the database storage component.
      - STR
    db.is.components: # List the IDs of the Integration Server database
components,
      - ISInternal # for which to manage the schemas in the database.
      - ISCoreAudit
      - DistributedLocking
    dbc.scripts:
      - DatabaseComponentConfigurator # List the product ID of Database Component
Configurator
      - PIEcdc # and the IDs of the database scripts for Integration
Server.
      - PIEEmbeddedCdc
      - PIEMobileCdc
    dbc.fixes: ALL # Database Component Configurator fixes to install.
layers:
```

```

db:
  productRepo: ${repo.product}      # Install products from this product repository.
  fixRepo:     ${repo.fix}          # Install fixes from this fix repository.
  templates:   dbc
  databases:
    storage:
      components: ${db.str.components}
      version:    latest
      db.type:    ${db.type}
      db.url:     ${db.admin.url}
      db.admin.username: ${db.admin.username}
      db.admin.password: ${db.admin.password}
      db.username: ${db.username}
      db.password: ${db.password}
      db.name:    ${db.name}
    schemas:
      products:  ${db.is.components.${version}}
      version:   ${db.version}
      db.type:   ${db.type}
      db.url:    ${db.url}
      db.username: ${db.username}
      db.password: ${db.password}
      db.name:   ${db.name}
  runtime:
    productRepo: ${is.repo.product}
    templates: is-server
templates:
  is-server:
    licenses:
      "IntegrationServer/config/licenseKey.xml": "${is.integrationserver.license.key}"

  fixes: ${is.fixes}
  products:
    integrationServer:
      ${is.instance.name}:
        primary.port: ${is.primary.port}
        diagnostic.port: ${is.diagnostic.port}
        jmx.port: ${is.jmx.port}
        license.file: IntegrationServer/config/licenseKey.xml
        db.url: jdbc:wm:${db.type}://${db.host}:${db.port};databaseName=${db.name}
        db.username: ${db.username}
        db.password: ${db.password}
        configuration:
          OSGI-IS_${instance.name}:
            ...
  dbc:
    products: ${dbc.scripts}
    fixes: ${dbc.fixes}
provision:
  default:
    db: ${nodes}
    runtime: ${nodes}

```

The `layers:` section defines a `db:` layer with the `dbc:` inline template. To manage a database, you must ensure that Database Component Configurator and the database scripts of the product(s) are installed in the product installation. The `dbc:` inline template in the example instructs Command Central to install Database Component Configurator and the database scripts for Integration Server, and then update them to the latest fix. The database scripts of the product are listed in `environments:default:dbc.scripts:`

The name of the database script starts with the product ID and ends with "cdc". You can find a list of the product database scripts in the sample database templates in [sagdevops-templates](#).

The `databases:` section of the `db:` layer includes two database action sections:

- `storage:` creates the database storage for the `database components:`, listed in the `environments:default:db.str.components:` parameter. This section includes the database parameters required to manage the storage component.
- `schemas:` identifies the database components for which Command Central manages the database schemas, and includes the database parameters required to manage the schemas. In the example, the database components for Integration Server are listed in the `environments:default:db.is.components:` parameter.

You can define a parameter that lists the database components for a product as follows:

```
environments:  
  default:  
    db.<productID>.components:  
      - dbcomponentA  
      - dbcomponentB
```

The sample database templates in [sagdevops-templates](#) include parameters for all products, which you can use as example. For usage notes and other details about the database components supported by Database Component Configurator, see *Database Component Descriptions and Installation Requirements in Installing Software AG Products*.

In the `provision:` section of the template, the database layer must map to a local or remote Platform Manager node. The database layer can only map to a remote Platform Manager node of release 9.10 or higher.

## How Does Command Central Apply a Template with a Database Layer?

Based on the value of the `environment.mode` argument that you include in the “[sagcc exec templates composite apply](#)” on page 451 command, Command Central does the following:

### **environment.mode=provision**

1. Installs Database Component Configurator and the database scripts listed in the `dbc.scripts:` parameter in the `environments:default:` section.

Note that when installing the database script of a product, Command Central installs all of its dependent scripts, even when the dependent scripts are not listed in the `dbc.scripts:` section.

2. Installs all available fixes for Database Component Configurator.
3. Creates the storage database component. The version of the database storage is always latest.
4. Executes the create action for the database schemas of the database components listed in the `db.productID.components:` parameter(s). Command Central can create database schemas with version equal or lower than the value of `db.version:` (which is the release version of the product).

In the example, Command Central creates the schemas for the database components listed in `db.is.components`:

5. Installs the product, creates and configures the product instance.

In the example, Command Central installs Integration Server, creates the `is.default` instance, updates the instance with the specified fix, and then configures the instance.

### **environment.mode=migration**

1. Installs Database Component Configurator and the database scripts listed in the `dbc.scripts` parameter in the `environments:default` section.

Note that when installing the database script of a product, Command Central installs all of its dependent scripts, even when the dependent scripts are not listed in the `dbc.scripts` section.

2. Installs all available fixes for Database Component Configurator.
3. Creates the storage database component. The version of the database storage is always latest.
4. Executes the migrate action for the database schemas of the database components listed in the `db.productID.components` parameter.

In the example, Command Central migrates the schemas for the database components listed in `db.is.components`:

5. Migrates the product instance.

In the example, Command Central migrates the `is.default` instance.

### **environment.mode=maintenance**

Use this mode to install a Database Component Configurator fix that contains database schema changes.

If you apply a maintenance template (a template that has a database layer and a run-time layer) in maintenance mode, Command Central does the following:

1. Pauses and then stops the run-time components on all nodes included in the `provision` section of the template.
2. Applies only the fixes defined in the database inline template and the run-time template. In the example, when Command Central processes the `dbc` inline template, it installs only the `dbc.fixes` to which the `fixes` section refers, but does not install the `dbc.scripts` to which the `products` section refers. When Command Central processes the `is-server` inline template, it installs only the `is.fixes` defined in the `fixes` section.

Note that in maintenance mode, Command Central does not create or migrate product instances, or apply configurations.

3. Executes the migrate database action for the database schemas of the database components listed in the `db.productID.components` parameter.

In the example, Command Central migrates the schemas for the database components listed in `db.is.components`:

If you apply a database template (a template that has only a database layer) in maintenance mode, you must stop the product instances on the nodes manually, before applying the template. Then after applying the database template, you must apply the fixes for the product instances in the installation.

By default, the run-time components in the installation are not started after applying a template with database layer in maintenance mode. If you want to control this behavior, include the `maintenance:` section in the template definition, as follows:

```
alias: is-maintenance-dcc
description: Update DCC to the latest fix
maintenance:
  options:
    pause: true | false      # "true" (default) - pause all runtimes, before stopping
                             them
                             # "false" - do not pause the runtimes
    shutdown: true | false  # "true" (default)- stop all runtimes before installing
                             fixes
                             # "false" - do not stop the runtimes
```

## Supported Database Component Configurator Parameters

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

Database Configurator Parameter	CLI or Composite Template Parameter
<code>dbms</code>	<code>db.type</code>
<code>component</code>	<code>component</code>
<code>product</code>	<code>product</code>
<code>version</code>	<code>version</code>
<code>user</code>	<code>db.username</code>
<code>password</code>	<code>db.password</code>
<code>dbname</code>	<code>db.name</code>
<code>url</code>	<code>db.url</code>
<code>adminuser</code>	<code>db.admin.user</code>
<code>adminpass</code>	<code>db.admin.password</code>
<code>tablespacedir</code>	<code>db.tablespace.dir</code>
<code>tablespacefordata</code>	<code>db.tablespace.data</code>

Database Configurator Parameter	CLI or Composite Template Parameter
tablespaceforindex	db.tablespace.index

The following table lists the Database Configurator parameters that are specified differently in the CLI and in the composite template:

This CLI argument	is defined in the composite template as
product	products
component	components

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 Template” on page 152](#).



# 11 Deploying Command Central Assets to Integration Cloud

---

■ About Deploying Command Central Assets to webMethods Cloud Container (Deprecated) .....	160
■ Defining Environment Variables for Platform Manager Assets .....	160
■ How Platform Manager Deploys Assets with Dependencies .....	161

## About Deploying Command Central Assets to webMethods Cloud Container (Deprecated)

---

Beginning with release 10.7, deploying Command Central assets to webMethods Cloud Container is deprecated. It is recommended to use Integration Server or Universal messaging to deploy assets and configurations to webMethods Cloud Container.

Command Central can export configuration properties for Software AG run-time components as YAML templates. Configuration properties are exported as Command Central composite assets using Software AG Designer, stored in the Landscape Asset Repository, and deployed to product instances, running on webMethods Cloud Container.

For information about the Software AG products you need to install for webMethods Cloud Container, see *Deploying to webMethods Cloud Container*.

To create Command Central assets and deploy them to webMethods Cloud Container, in Designer:

1. Connect to Command Central.
2. Select and export the configurations to be deployed.
3. Edit the YAML configurations.
4. Deploy the assets to webMethods Cloud Container.

For more information about deploying assets, see *Deploying to webMethods Cloud Container*.

## Defining Environment Variables for Platform Manager Assets

---

When deploying Software AG Platform Manager configuration assets, you must ensure that environment variables are resolved and applied correctly after Platform Manager is restarted. The following list gives you guidelines on how to define environment variables for Platform Manager assets. The guidelines apply for all supported platforms.

- Set global system environment variables at the operating system level. The ACDL file can resolve the global system environment variables when Platform Manager is restarted using the `.sh` | `.bat` or native service/daemon scripts. For example:

```
set ENV_VAR=BAR
profiles\SPM\bin\restart.bat
export ENV_VAR=BAR
profiles/SPM/bin/restart.sh
```

The ACDL file gets the value of BAR by referencing `$ENV_VAR`.

- Use Java system properties in place of environment variables for the scope of Platform Manager. You can set the Java system properties using the COMMON-JAVASYSPROPS configuration type. For example, you can set the "ENV\_VAR1" Java system property in a Command Central template:

```
COMMON-JAVASYSPROPS:
COMMON-JAVASYSPROPS: |
ENV_VAR1=BAR1
```

When Platform Manager is restarted, the ACDL file can get the value of BAR1 by referencing `$ENV_VAR1`.

- Set environment variables for the scope of Platform Manager by editing the `custom_wrapper.conf` file, located in the `PlatformManager_directory\profiles\SPM\configuration` directory. When Platform Manager is (re)started, the ACDL file can resolve the environment variable using the value specified in the `custom_wrapper.conf` file. For example, if you set the `ENV_VAR2` in the `custom_wrapper.conf` file:

```
set.ENV_VAR2=BAR2
```

The ACDL file can get the value of BAR2 by referencing `$ENV_VAR2`.

## How Platform Manager Deploys Assets with Dependencies

When deploying assets from an asset repository, Platform Manager determines the order in which to install the assets on the target environment based on the type of asset (configuration or product) and the asset dependencies. Platform Manager manages the dependencies between assets at the time it deploys an asset on the target installation.

When deploying product assets and Command Central configuration assets, Platform Manager always installs the configuration assets after it installs the product assets. For example, when Platform Manager deploys "packageA" (product asset for Integration Server) and "COMMON-JDBC" (configuration asset for Integration Server), Platform Manager first installs "packageA" and then applies the "COMMON-JDBC" configuration.

Platform Manager installs the dependent assets before it installs the main asset. For example, when "assetA" depends on "assetB", Platform Manager installs "assetB" before it installs "assetA".

Platform Manager installs all assets included in the repository deploy changeset and compares the dependent assets that it is about to deploy with the dependent assets already installed on the target installation by name. If a dependent asset is already installed on the target installation and is not included in the deploy changeset, Platform Manager installs the main asset, but does not attempt to re-install the dependent asset.

Note that if you want to re-install a dependent asset that is already installed on the target environment, you must include the dependent asset in the changeset.



# 12 Command Central Composite Assets

---

■	About Command Central Composite Assets .....	164
■	Adding Assets to the Build Source Directory .....	164
■	Setting Build Properties for Command Central Assets .....	164
■	Building Command Central Composite Assets .....	166

## About Command Central Composite Assets

---

Software AG Command Central exports configuration properties for Software AG run-time components as YAML templates, from which you can create composite assets. You can store the Command Central composite assets in a local or remote Landscape Asset Repository (LAR) and deploy them on Software AG run-time components running on-premises, or on webMethods Integration Cloud.

## Adding Assets to the Build Source Directory

---

You create Command Central configuration assets using the [“sagcc exec templates composite generate” on page 457](#) or the [“sagcc exec templates composite generate input” on page 465](#) CLI command.

To export the assets to the source directory that you intend to use for the asset build, use the [“sagcc get templates composite export” on page 478](#) CLI command.

When deploying assets to webMethods Integration Cloud, you can generate and export Command Central configuration assets in Software AG Designer. For more information about deploying Command Central assets to webMethods Integration Cloud, see the *webMethods Service Development Help* and the *Deploying to webMethods Integration Cloud* document.

## Setting Build Properties for Command Central Assets

---

### Important:

When you run the build script, if you specify a build property as an argument in the command line console, the value of the command argument overwrites the value of the property in the `build.properties` file.

You must set the following properties in the `build.properties` file in webMethods Asset Build Environment:

### **sag.install.dir**

Set the property to the directory in which webMethods Asset Build Environment is installed:

```
sag.install.dir=ABE_HOME
```

### **build.source.dir**

It is recommended that you structure the build source directory of the Command Central configuration assets as follows:

```
C:.\
|  <templates>
|  template-alias-1.yaml
|
|---template-alias-2
|     template.yaml
|
```

```

\---template-alias-3
|   template.yaml
|
|   \---files
|       license_key.xml

```

Each `template-alias` subdirectory contains the `template.yaml` file of the configuration asset and any other files required by the configuration properties included in the template.

The following naming conventions apply for the subdirectories and YAML files located directly under the “templates” directory:

- The name of the YAML file is the `alias:` specified in the YAML file.
- The name of each `template-alias` subdirectory is the `alias:` specified in the template YAML file located in the subdirectory.

With this build source directory structure, you can set the `build.source.dir` property as follows:

- To generate all Command Central assets available in the source directory:

```
build.source.dir=<root_path>/<templates>/
```

where `<templates>` is the name of the directory that contains the source template files. Command Central generates a composite asset for each:

- `<templates>/<template_alias>.yaml` file
- `<templates>/<template_alias>` subdirectory
- To generate just one of the available Command Central assets:
  1. Copy the template file (and any other required files) to a subdirectory directly under `<root_path>/<some_template_dir>`:

```
<root_path>/<some_template_dir>/<template_alias>/template.yaml
```

or copy the template file directly under `<root_path>/<some_template_dir>`:

```
<root_path>/<some_template_dir>/<template_alias>.yaml
```

2. Set the build source directory property to:

```
build.source.dir=<root_path>/<some_template_dir>/
```

## Example

In the following example, the configuration assets are located inside the “templates” build source directory. The name of the “IS-template” subdirectory matches the `alias: IS-template` in the `template.yaml` file, located in this subdirectory. The name of the `sag-template.yaml` file matches the `alias: sag-template` specified in the file.

```

|   templates
|   sag-template.yaml
|
|   \---IS-template
|       |   template.yaml

```

When `build.source.dir=<root_path>/templates/`, the composite assets generated after running the build script are:

```
| <build.output.dir>
|
\---CC
    |   IS-template.acdl
    |   IS-template.zip
    |   sag-template.acdl
    |   sag-template.zip
```

### **build.output.dir**

Set the property to:

```
build.output.dir=<root_path>/build/
```

The build script creates a subdirectory with name `CC` under `<root_path>/build` and places the Command Central composite assets from the output in that subdirectory.

### **enable.build.CC**

Set the property to `true`.

## **Building Command Central Composite Assets**

---

To run the build script to generate the Command Central composite assets from the source configuration assets (YAML templates), in a command line console, go to:

- `<root_path>/bin` and run the build script
- `<root_path>/master_build` and run `ant`
- `<root_path>/CC` and run `ant build.composites`

The output of the script is an ACDL descriptor file and a composite file for each template in the source directory.

The ACDL file is named `<template_alias>.acdl`, where `<template_alias>` is the alias of the template included in the composite file. The ACDL file lists:

- The ID of the product(s) on which to apply the configuration properties from the template YAML file included in the composite file.
- The configuration properties defined in the `environments:` section of the template YAML file.

The composite file is named `<template_alias>.zip`, where `<template_alias>` is the alias in the template YAML file. The `.zip` composite file contains either a single template YAML file or all files in a `<template_alias>` subdirectory, that is, the template YAML file and any files to which the template refers, such as the license key or SSH key files.

# 13 Using the Command Line Interface

---

■ Installing CLI as a Remote Client .....	169
■ Displaying Help for the Command Line Interface .....	169
■ Displaying the Version of the Command Line Interface .....	170
■ Executing Command Central Commands .....	170
■ Executing Platform Manager Commands .....	171
■ Getting Familiar with Using Commands .....	171
■ Return Codes from Command Execution .....	173
■ Invoking Commands from Scripts .....	173
■ Managing Database Components Using CLI Commands .....	188
■ Options for the Commands .....	193
■ Administration Commands .....	222
■ License Inventory Commands .....	226
■ Configuration Commands .....	229
■ Diagnostics Logs Commands .....	260
■ Instance Management Commands .....	266
■ Inventory Commands .....	273
■ Jobmanager Jobs Commands .....	301
■ Landscape Commands .....	306
■ License Keys Commands .....	325

- License Reports Commands ..... 334
- Lifecycle Commands ..... 348
- Monitoring Commands ..... 352
- Provisioning Bootstrap Installers Commands ..... 359
- Provisioning Fixes Commands ..... 364
- Provisioning Products Commands ..... 368
- Repository Commands ..... 372
- Resources Commands ..... 413
- Security Credentials Commands ..... 414
- Stacks and Layers Commands ..... 425
- Template Commands ..... 450

**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.
3. Open a command window as Administrator for Windows or a shell window for UNIX and run the .bat file or .sh script.

The following table lists the arguments you can use when running the bootstrapper and describes the values you provide for each argument.

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-10.7-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/cce
```

**Note:**

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

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

```
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.

## Displaying the Version of the Command Line Interface

---

From the command prompt, you can check the following details about the Command Central CLI:

- The CLI version.
- The location of the `/bin` directory.
- The location of the CLI configuration file.

To find the Command Central CLI details, enter `sagcc` and use the `{--version | -v}` option:

```
sagcc --version
```

## 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 215](#).

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 steps illustrate 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.

1. Use a `list` command to view the type of information the command returns.

For example, execute the following command to view a list of installations.

```
sagcc list landscape nodes
```

The output includes alias names for all the installations. You can use the alias names in subsequent commands to get data for an installation, update an installation, execute actions against an installation, or delete an installation.

2. Use a `get` or `list` command to retrieve information for a specific instance.

#### Note:

The `get` and `list` commands are equivalent.

For example, assume the `list` command provided information for an installation that has the alias name “`sag01`”. To retrieve information for the “`sag01`” installation, returning the information to an output file in XML format, execute the following command:

```
sagcc get landscape nodes sag01
--output info --format xml
```

3. Use a `create` command to create a new instance.

You can edit the output file that a `get` command returns to specify the information for the new instance. Then you can use that file as input to the `create` command.

For example, to create a new installation with alias name “`new`”, edit the `info.xml` file that the `get` command returned to supply the alias name, URL, and description for the new installation. Then execute the following command:

```
sagcc create landscape nodes
--input info.xml
```

**Note:**

If you execute the `list` command again, the command lists the “`new`” installation.

4. Use an `update` command to update data for an instance.

You can use a `get` command to retrieve information for the specific instance you want to update, returning the output to a file. Then you can update the output file the `get` command returns and use that as the input to the `update` command.

For this example, update the “`new`” installation. Execute the following command to retrieve information for the “`new`” installation, returning the output to a file in XML format:

```
sagcc get landscape nodes new
--output updatefile --format xml
```

Update the data in the returned “`updatefile`”. For example, you might specify a new description. Then execute the following command to update the installation information:

```
sagcc update landscape nodes new
--input updatefile
```

5. Use an `exec` command to execute an action against an instance.

To generate a new ID for the “`new`” installation, execute the following command:

```
sagcc exec landscape nodes new
generateNodeId
```

6. Use a `delete` command to remove an instance.

To delete the “`new`” installation, execute the following command:

```
sagcc exec landscape nodes new
```

**Note:**

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

## 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 that 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.
4	Indicates that the execution of the command failed with an error. A command returns 4 when the HTTP response code is greater or equal to 400 and less than 500.
5	Indicates that the execution of the command failed with an error. A command returns 5 when the HTTP response code is greater or equal to 500 and less than 600.
10	Indicates that the output that a command returned does not match the expected values specified with the <code>{--expected-values  -e}</code> option.

## Invoking Commands from Scripts

### 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.

## Parameter and Description

server

Optional. Identifies the Command Central or Platform Manager server on which to execute the command. For example:

```
server="https://localhost:8092/spm"
```

The following lists the order used to determine the value used for the server:

1. Value set with the `cc` task.
2. Value set with the `ccsetup` task.
3. Value `https://localhost:8090/cce`

username

Optional. Specifies the user name to use for authentication. For example:

```
username="Administrator"
```

The following lists the order used to determine the value used for the user name:

1. Value set with the `cc` task.
2. Value set with the `ccsetup` task.
3. Value defined in the `CC_USERNAME` environment variable.
4. "Administrator"

trustAllHosts

Optional. Specifies whether to trust all hosts. When the parameter is included, Command Central does not verify the name of the server host. For example:

```
trustAllHosts="true"
```

The following lists the order used to determine the value for the truststore file:

1. Value set in the custom `cc.properties` file located in the `user_home\.sag` directory.
2. Value set in the Command Central default `cc.properties` file located in the `CC_CLI_HOME\conf` directory.

sslTruststoreFile

Optional. Specifies the location of the truststore file. For example:

```
sslTruststoreFile=="${cce.cli.truststore.file.location}"
```

The following lists the order used to determine the value for the truststore file:

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

### Parameter and Description

- 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

### 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.

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.

### Parameter and Description

accept

Deprecated. Optional. Use *outputFormat* in place of *accept*.

Specifies the format for the returned data. You supply a content type with the *accept* parameter that is used on the HTTP Accept request header sent to Command Central or Platform Manager. For example:

```
accept="json"
accept="xml"
accept="csv"
accept="tsv"
```

If you omit the *accept* parameter, *xml* is used.

### Note:

Use either the *accept* or the *format* parameter to specify the format of the returned data. If you specify both, the value you specify with the *accept* is used.

checkevery

## Parameter and Description

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 193](#). 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

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

### Parameter and Description

---

- 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"
```

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.

---

## Parameter and Description

`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 `errorproperty` parameter, 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`.

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.

## Parameter and Description

### 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:

```
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 `outputFormat`.

## Parameter and Description

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

### 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

## Parameter and Description

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.

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"
```

## Parameter and Description

- 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.

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.

**Parameter and Description**

---

**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.

---

`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"
```

---

## Parameter and Description

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.

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

---

`server`

### Parameter and Description

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

## Parameter and Description

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.

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
```

---

## Managing Database Components Using CLI Commands

### Installing Database Scripts for Software AG Products

Before you create a database component for a product with version 10.5, you must install the database scripts for that product. You can install database scripts using the [“sagcc exec provisioning products install” on page 368](#) command. In the artifacts argument of the command, list the IDs of the product database scripts that you want to install.

#### Example

To install the database scripts with IDs “PIEcdc” and “MWScdc”, for Integration Server and My webMethods Server respectively, from a master repository with name “webMethods-EUR” on a remote installation with alias “sag01”:

```
sagcc exec provisioning products sag01 webMethods-EUR install artifacts=PIEcdc,MWScdc
```

### 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 36](#).

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"
```

## 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 for local or remote nodes with version 10.3 or lower:

```
sagcc list administration product <node> DatabaseComponentConfigurator
database migrations db.type=db_type
{product|component}=productID|componentID
```

## Example

To list the supported migration versions for a remote node with alias "sag01\_103" on the SQL server database for the "MWS" run-time component:

```
sagcc list administration product sag01_103 DatabaseComponentConfigurator
database migrations db.type=sqlserver component=MWS
```

## Options for the Commands

### Common Options

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

- ["accept" on page 194](#)
- ["check-every" on page 196](#)
- ["configuration-file" on page 197](#)

- “debug” on page 197
- “error” on page 198
- “expected-values” on page 200
- “force” on page 202
- “format” on page 202
- “input” on page 204
- “input-format” on page 206
- “log” on page 207
- “media-type” on page 208
- “output” on page 209
- “output-format” on page 210
- “password” on page 212
- “properties” on page 213
- “quiet” on page 214
- “server” on page 215
- “ssl-truststore-file” on page 216
- “ssl-trust-all-hosts” on page 217
- “ssl-truststore-password” on page 217
- “username” on page 219
- “wait” on page 220

**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`.

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"> <li>■ <code>application/xml</code></li> <li>■ <code>application/json</code></li> <li>■ <code>application/vnd.sagcc.asset+zip</code></li> <li>■ <code>application/yaml</code></li> <li>■ <code>text/plain</code></li> <li>■ <code>text/tab-separated-values</code></li> <li>■ <code>text/csv</code></li> </ul>

## 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 204](#).
- 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 209](#).

## 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>	Specifies the number of seconds the command waits before checking for expected output specified by the <code>{--expected-values   -e}</code> option.  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.

### 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 200](#).

## 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><b>Note:</b> 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 [“Update the SSL Connection Settings for the CLI” on page 502](#).

### 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 Alias Name Status Url Host Url Port
Output 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"> <li>■ Absolute directory path and filename.</li> <li>■ Relative directory path and filename. The path is relative from where you initiated the command.</li> <li>■ Filename of a file in the same directory where you initiated the command.</li> </ul>

## 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> 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> .

### 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 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.  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.

Operator	Description
&	Indicates that the value includes a logical AND operator. For example, <code>RUNNING&amp;RESPONSIVE</code> checks if the output contains both <code>RUNNING</code> and <code>RESPONSIVE</code> .
	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> : <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 <a href="#">“properties” on page 213</a> .
<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 <a href="#">“properties” on page 213</a> .
<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 194](#).
- 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 204](#).
- 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 209](#).
- 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 213](#).

## 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"> <li>■ An <code>.xml</code> file containing input data in XML format</li> <li>■ A <code>.json</code> file containing input data in JavaScript Object Notation format</li> <li>■ A <code>.properties</code> file containing input data in key/value pairs format.</li> </ul> <p>When identifying the input file, you can specify:</p> <ul style="list-style-type: none"> <li>■ Absolute directory path and filename.</li> <li>■ Relative directory path and filename. The path is relative from where you initiated the command.</li> <li>■ Filename of a file in the same directory where you initiated the command.</li> </ul>

## 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 table 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"><li>■ <code>application/xml</code>   <code>xml</code></li><li>■ <code>application/json</code>   <code>json</code></li><li>■ <code>text/plain</code>   <code>text</code></li><li>■ <code>text/tab-separated-values</code>   <code>tsv</code></li><li>■ <code>text/csv</code>   <code>csv</code></li></ul> <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 <code>text/plain</code>.</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>	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: <ul style="list-style-type: none"> <li>■ Absolute directory path and filename.</li> <li>■ Relative directory path and filename. The path is relative from where you initiated the command.</li> <li>■ Filename of a file in the same directory where you initiated the command.</li> </ul>

## 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
<i>content-type</i>	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
<i>file</i>	Specifies the file where you want the output written. If the file you specify does not exist, the command creates it. You can specify: <ul style="list-style-type: none"> <li>■ Absolute directory path and filename.</li> <li>■ Relative directory path and filename. The path is relative from where you initiated the command.</li> <li>■ Filename of a file in the same directory where you initiated the command.</li> </ul>

### 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"><li>■ <code>application/xml   xml</code></li><li>■ <code>application/json   json</code></li><li>■ <code>application/vnd.sagcc.asset+zip</code></li><li>■ <code>application/yaml</code></li><li>■ <code>application/properties</code></li><li>■ <code>text/plain   text</code></li><li>■ <code>text/tab-separated-values   tsv</code></li><li>■ <code>text/csv   csv</code></li></ul> <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 <code>text/tab-separated-values</code>.</p> <p>For the content types that a command supports, see the documentation for a specific command.</p>

---

### Example

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

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

### Recommendations When Selecting a Content Type

Use the information in this topic to determine which of the output content types supported by Command Central to use with different types of clients. The following table specifies the recommended output format for each available client type and describes the considerations, related to each format.

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	XML, JSON	<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	TSV, CSV	<p>The TSV or CSV content types are well-supported by UNIX (and to some extent Windows) command line shells.</p> <div data-bbox="717 1234 1461 1402" style="background-color: #f0f0f0; padding: 5px;"> <p><b>Note:</b> New command line shells, for example Windows PowerShell, can also process the XML and JSON content types.</p> </div> <p>With the TSV and CSV content types, the property names remain unchanged, but the order and the set of properties may change.</p> <p>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>The scripting clients can also remove the header line, using the <code>includeHeaders=false</code> parameter.</p>

Type of client	Recommended format	Considerations
CLI end user	TSV	<p><b>Tip:</b> 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> <p>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>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>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.
- You can use the [“sagcc exec security encrypt” on page 420](#) command to encrypt 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
[includeHeaders={labels   properties   none}]	<p>Specifies whether you want the output to include a header line. Specify one of the following:</p> <ul style="list-style-type: none"> <li>■ <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.</li> <li>■ <code>properties</code> to include a header line containing the property key name for each field. For example “product.version” might be the display name if the output includes the version of a product.</li> <li>■ <code>none</code> to omit headers from the output.</li> </ul>
[properties=keys]	<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
<code>[count]</code>	Optional. Specifies the number of times to resubmit a command.  If you omit the <code>count</code> 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 218](#) and [“wait-for-cc” on page 221](#).
- 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 451](#).
- 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 176](#).

## 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"> <li>■ When specifying the URL of a Command Central server, if you omit: <ul style="list-style-type: none"> <li>■ Protocol, the command uses “https://”.</li> <li>■ Port number, the command uses “8090”, which is the default port for a Command Central server.</li> <li>■ Context, the command uses “cce”.</li> </ul> </li> </ul>

Argument	Description
	<ul style="list-style-type: none"> <li>You must always provide the full URL, including the protocol, when identifying a Platform Manager server.</li> </ul>

## 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>	<p>Specifies the fully qualified path to the truststore location.</p> <p>The default Command Central truststore with name <code>demo-truststore.jks</code> is located in the <i>Software AG_directory</i> \CommandCentral\client\conf directory.</p> <p><b>Note:</b></p>

Argument	Description
	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.
- You can use the [“sagcc exec security encrypt” on page 420](#) command to encrypt the password.

## 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 221](#).
- 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 451](#).

- 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 176](#).
- Running a command with `{--sync-job | -j}` and `includeHeaders=none` does not return job result and progress details. To obtain job result and progress details, do use `{--sync-job | -j}` together with `includeHeaders=none` in the same command.

## 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 212](#)

### 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 200](#).

## 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
[seconds]	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 218](#).
- 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 176](#).

### Examples

- To wait for 60 seconds for Command Central to come online:

```
--wait-for-cc 60
```

## Administration Commands

---

### 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.
<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 <a href="#">“Common Options” on page 193</a> .

## 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.

Argument or Option	Description
[ <i>namespace</i> ]	Optional. The namespace identifier for the administrative action. The namespace groups together administrative functions that relate to the same domain, such as migration or configuration.  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.
[ <i>administration_action</i> ]	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.
[ <i>argName1=value1</i> ] [ <i>argName2=value2</i> ]...	Optional. The arguments of the administrative action.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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.

## License Inventory Commands

### 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"> <li>■ <i>LocationId</i></li> <li>■ <i>ContractId</i></li> <li>■ <i>ContractItemId</i></li> <li>■ <i>BundleComponentId</i></li> </ul>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

#### 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"> <li>■ <i>LocationId</i></li> <li>■ <i>ContractId</i></li> <li>■ <i>ContractItemId</i></li> <li>■ <i>BundleComponentId</i></li> </ul>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

### 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
```

## Configuration Commands

### Return Statuses from Executing Configuration Commands

The following table lists the return statuses that can result from executing a CLI configuration command on Command Central or Platform Manager.

Return Status	Description
CURRENT	The command is executed successfully.
DELETED	The configuration instance is deleted successfully.
NOT_DELETED	The configuration instance is not deleted.
PENDING_RESTART	Manual restart required. You should restart the run-time component, that hosts the configuration instance you are creating, updating, or deleting for the changes to become effective.
PENDING_SCHEDULED_RELOAD	The configuration changes will get reloaded in the product instance, typically after a pre-set period of time, without restarting the product instance.
UNKNOWN	Command Central cannot determine the action to execute after applying the configuraiton changes. For more

Return Status	Description
	information, read the Command Central and Platform Manager logs.

## Adding Configurations Exported from an Installed Product Instance to an Input File

You can export a configuration from an installed product instance, managed by Command Central and use the exported configuration data in the input file when creating or updating a configuration instance of the same configuration type.

1. In the Command Central web user interface, go to **Environments > ALL > Instances** and click the instance, from which you want to export configuration data.
2. Go to the Configuration tab and select the required configuration type from the drop-down box.
3. Click the name of the required configuration in the table.
4. Click **Export**.
5. Select **Configuration data only** and select the required format from the **Data format** drop-down box.
6. Copy the output and paste it in the input file you intend to use with the create or update configuration command.

## 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"> <li>■ CommonSettings.xsd</li> <li>■ EmailSettings.xsd</li> <li>■ JDBCSettings.xsd</li> <li>■ KeystoreTruststoreSettings.xsd</li> <li>■ LicenseLocation.xsd</li> <li>■ log4j.xsd</li> <li>■ PortSettings.xsd</li> </ul>
<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 <a href="#">“Common Options” on page 193</a>.</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 Adminisrator --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>	Required. Specifies the configuration type to compare.  You can determine the IDs for configuration types using <a href="#">sagcc list configuration types</a> .
<code>runtimeComponentInfoId=id1</code> <code>runtimeComponentInfoId=id2</code> <code>[runtimeComponentInfoId=id3 ...</code> <code>runtimeComponentInfoId=idn]</code>	Required. Specifies the information IDs of two or more run-time components that you want to compare.

Argument or Option	Description
	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 <a href="#">sagcc list landscape nodes</a> . You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 are configuring a new port for an Integration Server, you can use this command to supply the data for the configuration type COMMON-PORTS and 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]
[/--input-format | -m} content-type]
[/--password | -p} password]
[/--quiet | -q]
[/--server | -s} url]
[/--sync-job | -j]
[/--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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</p>
<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 <a href="#">sagcc list configuration types</a>.</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</p>

Argument or Option	Description
<pre>{--input   -i} file{.xml  .json .properties}</pre>	<p>whole landscape. For more information, see <a href="#">sagcc add security credentials sharedsecret</a>.</p> <p>Required. Identifies an input file that contains the configuration data. For more information, see <a href="#">“input” on page 204</a>.</p> <div data-bbox="764 474 1474 709" style="background-color: #f0f0f0; padding: 5px;"> <p><b>Note:</b> Some of the COMMON-* configuration types do not support all input file types (.xml, .json, or .properties) when creating a configuration instance. Although not specifically supported, if you use plain text, the server attempts to convert the data into a supported format.</p> </div>
[options]	<p>Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</p>

## 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 [Administering Software AG Products Using Command Central](#).
- 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 are creating. 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 `--format xml` option to retrieve the data for an existing COMMON-PORTS instance in XML format.
- The data in the input file must match the expected schema for the configuration type. You can use [sagcc exec configuration validation create](#) to validate input data that you want to use to update the configuration instance.
- You can retrieve schemas for common configuration types from a Platform Manager server and use the schemas to validate an XML input data file. To access the schemas, navigate to:

```
http(s)://hostname:port/spm/configuration/common/
```

Where *hostname* is the host name of the required Platform Manager server and *port* is the port number of the Platform Manager installation.

For example, to retrieve the log4j schema from a Platform Manager with host name “rubicon2” and port “8092”, navigate to:

```
https://rubicon:8092/spm/configuration/common/log4j.xsd
```

You can also use the [sagcc get configuration common](#) command to retrieve schemas for common configuration types.

- You can export a configuration from an installed product instance, managed by Command Central and use the exported configuration data in the input file. For more information, see [“Adding Configurations Exported from an Installed Product Instance to an Input File”](#) on page 230.
- For information about the return statuses of the command, see [“Return Statuses from Executing Configuration Commands”](#) on page 229.
- 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 215 and [“username”](#) on page 219. 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 219](#). The command specifies “secret” for the user’s password.

## sagcc delete configuration data

Deletes a specified configuration instance from a specified run-time component. For example, you can delete a previously created port on Integration Server that you no longer require.

### 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]
[{--sync-job | -j}]
[{--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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance you want to delete.</p>

Argument or Option	Description
	You can determine the IDs for configuration instances using <a href="#">sagcc list configuration instances</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- You cannot delete some configuration data. The command returns an error if you attempt to perform an unsupported operation. In most cases, if you can create the configuration instance, you can delete it. You can verify if you can delete a configuration instance by running [sagcc exec configuration validation delete](#).
- For information about the return statuses of the command, see [“Return Statuses from Executing Configuration Commands” on page 229](#).

## 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 215](#).

## 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>	<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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list configuration instances</a>.</p>

Argument or Option	Description
[sharedsecret= <i>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 <a href="#">sagcc add security credentials sharedsecret</a>. 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"> <li>■ 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 <a href="#">sagcc add security credentials sharedsecret</a>.</li> <li>■ If the command is executed on Platform Manager, Platform Manager does not include the passwords that the configuration contains in the output.</li> </ul>
[ <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 <a href="#">“Common Options” on page 193</a>.</p>

## 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 219 and “[password](#)” on page 212.

## sagcc update configuration data

Updates the data for a specified configuration instance that belongs to a specified run-time component. For example, you can 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
[--input-format | -m] content-type
[--log | -l] file
[--output-format | -m] content-type
[--password | -p] password
[--quiet | -q]
[--server | -s] url
[--sync-job | -j]
[--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 <a href="#">sagcc list landscape nodes</a>.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component to which the instance you want to update belongs.</p> <p>You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list configuration instances</a>.</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 <a href="#">sagcc add security credentials sharedsecret</a>.</p>
<i>--input   -i</i> <i>filename{.xml   .json   .properties}</i>	<p>Required. Identifies an input file that contains the updated configuration data. For more information, see <a href="#">“input” on page 204</a>.</p>
	<p><b>Note:</b> Some of the COMMON-* configuration types do not support all input file types (.xml, .json, or .properties) when creating a configuration instance. Although not</p>

Argument or Option	Description
	specifically supported, if you use plain text, the server attempts to convert the data into a supported 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 <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- To determine how to specify the data in the input file, use [sagcc get configuration data](#) to retrieve data for the configuration instance you are updating. 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 `--format xml` option to retrieve the data for the COMMON-PORTS instance in XML format.
- 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 from a Platform Manager server and use the schemas to validate an XML input data file. To access the schemas, navigate to:
 

```
http(s)://hostname:port/spm/configuration/common/
```

Where *hostname* is the host name of the required Platform Manager server and *port* is the port number of the Platform Manager installation.

For example, to retrieve the log4j schema from a Platform Manager with host name “rubicon2” and port “8092”, navigate to:

```
https://rubicon:8092/spm/configuration/common/log4j.xsd
```

You can also use the [sagcc get configuration common](#) command to retrieve schemas for common configuration types.
- You can export a configuration from an installed product instance, managed by Command Central and use the exported configuration data in the input file. For more information, see [“Adding Configurations Exported from an Installed Product Instance to an Input File” on page 230](#).
- 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.
- For information about the return statuses of the command, see [“Return Statuses from Executing Configuration Commands” on page 229](#).

- You can specify the content type for the output data of 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 210](#).

If you do not specify the content type for the output data, the command returns the default output format, XML.

## 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 215](#).

## 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 219](#). 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>	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 <a href="#">sagcc list landscape nodes</a>.</p>
<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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list configuration instances</a>.</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 <a href="#">“Common Options” on page 193</a>.</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 219](#) and [“password” on page 212](#).

## 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>	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 <a href="#">sagcc list landscape nodes</a>.</p>
<i>componentid</i>	<p>Required. Specifies the ID of the run-time component for which you want to list configuration instances.</p> <p>You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a>.</p>
[ <i>instanceid</i> ]	<p>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.</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 <a href="#">“Common Options” on page 193</a>.</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 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list configuration types</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## 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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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-time component identified by the <i>componentid</i> argument.</p>

Argument or Option	Description
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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> <p>You can determine the IDs for configuration types using <a href="#">sagcc list configuration types</a>.</p>

Argument or Option	Description
	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.
<pre>{--input   -i}   filename{.xml   .json .properties}</pre>	Required. Identifies an input file that contains the configuration data to validate. For more information, see <a href="#">“input” on page 204</a> .
	<p><b>Note:</b> 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><b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get configuration data</a> 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 <a href="#">sagcc get configuration data</a> with the <code>--format xml</code> option to retrieve the data for an existing COMMON-PORTS instance in XML format.</p>
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). 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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list configuration instances</a>.</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 <a href="#">“Common Options” on page 193</a>.</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 215](#).

## 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</p>
<i>instanceid</i>	<p>Required. Specifies the ID of the instance.</p> <p>You can determine the IDs for configuration instances using <a href="#">sagcc list configuration instances</a>.</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 <a href="#">“input” on page 204</a>.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p><b>Note:</b> 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><b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get configuration data</a> 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 <a href="#">sagcc get configuration data</a> with the <code>--format xml</code> option to retrieve the data for an existing COMMON-PORTS instance in XML format.</p> </div>
[ <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 <a href="#">“Common Options” on page 193</a>.</p>

## 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 215](#).

## 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 219](#). The command specifies “secret” for the user’s password.

## Diagnostics Logs Commands

### 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>	<p>Required. Only for Command Central. Specifies the alias name of the Platform Manager that manages the run-time component for which you want to retrieve information.</p> <p>You can determine the alias name of the Platform Manager node by using <a href="#">sagcc list landscape nodes</a></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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">sagcc list diagnostics logs</a>.</p>
{full   tail   head}	<p>Required. Identifies the log entries you want to retrieve.</p> <ul style="list-style-type: none"> <li>■ Specify <code>full</code> to retrieve all log entries.</li> <li>■ Specify <code>tail</code> to retrieve the most recent log entries from the end of the log file.</li> <li>■ Specify <code>head</code> to retrieve entries from the beginning of the log file.</li> </ul>
[lines=number]	<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=number</code> the command returns 100 entries.</p>
[regex=expression]	<p>Optional. Narrows the retrieved log entries to those that meet the specified regular expression.</p>
[search=text]	<p>Optional. Narrows the retrieved log entries to those that contain the specified search text.</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 <a href="#">“Common Options” on page 193</a>.</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 219](#). 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 219](#). 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 <a href="#">sagcc list inventory components</a> .
<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 <a href="#">sagcc list diagnostics logs</a> .
<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 <a href="#">“output” on page 209</a> .
<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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">sagcc list inventory components</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## 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
```

## Instance Management Commands

---

### 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 <a href="#">sagcc list landscape nodes</a>.</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 more information, see <a href="#">“input” on page 204</a>.</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>
[ <i>options</i> ]	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</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 <a href="#">sagcc list landscape nodes</a> .
<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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">sagcc list landscape nodes</a> .
<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 <a href="#">“Common Options” on page 193</a> .

### 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 <a href="#">sagcc list landscape nodes</a> .
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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>	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 <a href="#">sagcc list landscape nodes</a> .
<i>componentid</i>	Required. Specifies the ID of the run-time component for which you want to update configuration properties.
[ <i>key=value</i> ]	Optional. A list of properties that describe the elements of the new instance, such as name and port settings. The properties included in this list are product specific.
[-i [ <i>file</i> {.xml .json .properties}]]	Optional. Identifies an input file that contains the new configuration data for the run-time component. For more information, see <a href="#">“input” on page 204</a> .  For the correct format of an XML properties file, see the Properties class in the Oracle Java Platform Standard Edition API specification.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### 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
```

## Inventory Commands

### 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:

```
sagcc list inventory assets nodeAlias runtimeComponentId
[options]
```

- Platform Manager syntax:

```
sagcc list inventory assets runtimeComponentId [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 <a href="#">sagcc list landscape nodes</a> .
<i>runtimeComponentId</i>	Required. Specifies the ID of the run-time component for which you want to retrieve asset information.

Argument or Option	Description
	You can view a list of run-time components and their IDs using <a href="#">“sagcc list inventory components” on page 279</a> .
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Example When Executing on Command Central

To list the assets for the run-time component with ID "OSGI-CCE" installed in the local installation and save the information to an XML file with name "assets":

```
sagcc list inventory assets local OSGI-CCE --format xml --output assets
```

### Example When Executing on Platform Manager

To list the assets for the run-time component with ID MwsProgramFiles-default 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 MwsProgramFiles-default --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
<i>node_alias</i>	<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 <a href="#">sagcc list landscape nodes</a>.</p>
<i>componentid</i>	<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 <a href="#">sagcc list inventory components</a>.</p>
[ <i>attribute=value</i> ]	<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>
[{--input   -i} filename{.xml .json}]	<p>Optional. Identifies an input file that contains the data for the new search attribute.</p> <p>You retrieve attribute data for the .xml   json file using the <code>sagcc get inventory components attributes</code> command:</p> <p>For more information, see <a href="#">“input” on page 204</a>.</p>
[ <i>options</i> ]	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</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 351](#). For information about including search criteria in search inventory commands, see [“Specifying Search Criteria for Inventory Commands” on page 299](#).

## 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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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 <a href="#">“Common Options” on page 193</a>.</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/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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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>
[ <i>options</i> ]	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</p>

## 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 219](#). 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]	Command Central only.  Optional. Specifies the alias name of the installation for which you want to retrieve component information.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .  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.
[componentid]	Optional. Specifies the ID of the run-time component for which you want to retrieve information.

Argument or Option	Description
	You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
[ <i>criteria</i> ]	Command Central only.  Optional. Narrows down the list of returned run-time components to only those that match the search criteria you specify. For more information, see <a href="#">“Specifying Search Criteria for Inventory Commands”</a> on page 299.
[ <i>start=number</i> ]	Command Central only.  Optional. Limits the results the command returns those starting at specified number in the results.  For example, if you want to return information for the 5th through the 8th run-time components in the results, use <code>start=5 size=4</code> .
[ <i>size=number</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 run-time component.
[ <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 <a href="#">“Common Options”</a> on page 193.

## 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 215](#) and [“username” on page 219](#). 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 215](#) and [“username” on page 219](#). 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 215](#) and [“username” on page 219](#). 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
```

## 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
<i>node_alias</i>	<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 <a href="#">sagcc list landscape nodes</a>.</p>
<i>componentid</i>	<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 <a href="#">sagcc list inventory components</a>.</p>
[ <i>options</i> ]	<p>Optional. The command supports all options supported by Command Central. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</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 219](#). 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>	Required. Specifies the alias name of the installation for which you want to retrieve component information.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
<i>componentid</i>	Required. Specifies the ID of the run-time component that you want to update.  You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
{--input   -i} <i>filename</i> {.xml .json .properties}	Required. Identifies an input file that contains the updated data for the run-time component. For more information, see <a href="#">“input” on page 204</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#), [“username” on page 219](#), and [“password” on page 212](#).

## 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</p>
[attribute=value]	<p>Optional. The name of the search attribute and the new matching value for the property.</p>
{--input   -i} filename{.xml .json}	<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 <a href="#">“input” on page 204</a>.</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 <a href="#">“Common Options” on page 193</a> .

## 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 219](#). 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>	Required. Specifies the alias name of the installation for which you want to delete component information.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
<i>componentid</i>	Required. Specifies the ID of the run-time component for which you want to delete a search attribute.  You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
[attribute]	Optional. The name of the search attribute that you want to delete.
[options]	Optional. The command supports all options supported by Command Central. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 219](#). 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>	<p>Command Central only.</p> <p>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.</p> <p>You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a>.</p>
<i>fixId</i>	<p>Required. Specifies the fix ID of the fix for which you want to retrieve product information.</p> <p>You can determine the IDs for fixes using “<a href="#">sagcc list inventory fixes</a>” on page 290.</p>

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 <a href="#">“Common Options” on page 193</a> .

### 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/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 219](#). 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 219](#). 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 <a href="#">sagcc list landscape nodes</a> .
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 219](#). 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> ]	<p>Command Central only.</p> <p>Optional. Specifies the alias name of the installation for which you want to retrieve fix information.</p> <p>You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a>.</p> <p>If you do not specify an alias name nor search criteria, the command lists all fixes in all installations that Command Central manages.</p>
[ <i>searchCriteria</i> ]	Command Central only.

Argument or Option	Description
	Optional. Narrows down the list of returned fixes to only those that match the search criteria you specify. For more information, see <a href="#">“Specifying Search Criteria for Inventory Commands”</a> on page 299.
[start= <i>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> .
[size= <i>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 <a href="#">“Common Options”</a> on page 193.

## 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 219. 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 215](#) and [“username” on page 219](#). 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 215](#) and [“username” on page 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory products</a>.</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 <a href="#">“Common Options” on page 193</a>.</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 219](#). 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 219](#). 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
nodeAlias= <i>alias1</i> nodeAlias= <i>alias2</i> [nodeAlias= <i>alias3</i>	Required. Specifies the alias names of two or more installations that you want to compare.

Argument or Option	Description
... <i>nodeAlias=aliasn</i>	You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
<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 <a href="#">“Common Options” on page 193</a> .

## 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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">“Specifying Search Criteria for Inventory Commands” on page 299</a>.</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> <p>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>.</p>
[ <i>size=count</i> ]	<p>Command Central only.</p>

Argument or Option	Description
	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.
<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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). 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 215](#) and [“username” on page 219](#). 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
```

## 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

The following table lists the attribute names you can use as a search criteria for each inventory command.

Command	Attribute Names
sagcc list inventory components	■ nodeName
	■ nodeAlias
	■ nodeUrl
	■ environmentName
	■ environmentAlias
	■ runtimeComponentInfoId
	■ runtimeComponentId
	■ runtimeComponentDisplayName
	■ runtimeComponentProductId
	■ runtimeComponentCategory
■ runtimeComponentRuntimeParentId	

Command	Attribute Names
<hr/> <code>sagcc list inventory fixes</code>	You can combine any of the pre-defined attribute names in this list with new search attributes added using <a href="#">sagcc create inventory components attributes</a> .
	■ nodeName
	■ nodeAlias
	■ nodeUrl
	■ environmentName
	■ environmentAlias
	■ fixId
	■ fixDisplayName
	■ fixVersion
	■ fixGroup
■ fixProducts	
<hr/> <code>sagcc list inventory products</code>	■ nodeName
	■ nodeAlias
	■ nodeUrl
	■ environmentName
	■ environmentAlias
	■ productId
	■ productCanonicalId
	■ productDisplayName
	■ productParentId
	■ productGroup
	■ productProfileDir
	■ productCode
	■ productVersion
■ productInstallTime	

## Specifying the Value

When specifying the value, you can include the \* pattern-matching character to match multiple characters 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
```

## Jobmanager Jobs Commands

### 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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 219](#) and [“password” on page 212](#).

## 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 219](#) and [“password” on page 212](#).

## 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 <a href="#">“Common Options” on page 193</a> .

### 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
```

## Landscape Commands

### 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.
nodes	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.

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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#), [“username” on page 219](#), and [“password” on page 212](#).

- 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=env_alias</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=name]</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=description]</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} filename{.xml .json}]</code>	<p>Optional. Identifies an input file that contains the data for the new environment. For more information, see <a href="#">“input” on page 204</a>.</p> <p><b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get landscape environments</a> to retrieve data for an existing environment. For example, if you want to use an XML input file, use <a href="#">sagcc get landscape environments</a> with the <code>--format xml</code> option to retrieve the data in XML format.</p>

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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). 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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#).

## 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
<i>env_alias</i>	Required. Specifies the alias name of the environment whose information you want to retrieve.
[ <i>nodes</i> ]	Optional. Indicates you want the command to return the information about the installations in the environment.  If you omit the <i>nodes</i> parameter, the returned information will not include the list of installations in 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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.
nodeAlias= <i>alias1</i> [nodeAlias= <i>alias2</i> ... nodeAlias= <i>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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). 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
<code>env_alias</code>	Required. Specifies the alias name of the environment whose description you want to update.
<code>[name=name]</code>	Optional. Specifies the updated display name for the environment.  If you use a value that includes spaces, place quotes around the value, for example: <pre>name="Dev Environment"</pre>
<code>[description=description]</code>	Optional. Specifies the updated description for the environment.  If you use a value that includes spaces, place quotes around the value, for example: <pre>description="A description with spaces"</pre>
<code>[{--input   -i} filename{.xml .json}]</code>	Optional. Identifies an input file that contains the updated data for the environment. For more information, see <a href="#">“input” on page 204</a> .  <b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get landscape environments</a> to retrieve data for the environment you want to update. For example, if you want to use an XML input file, use <a href="#">sagcc get landscape environments</a> 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 <a href="#">“Common Options” on page 193</a> .

## 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 215 and “[username](#)” on page 219. 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]
[installationType={Production|Development|Test}]
[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
[name= <i>name</i> ]	<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> <hr/> <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="732 506 1459 541">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>
[description= <i>description</i> ]	<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="732 789 1459 825">description="A description with spaces"</pre>
[installationType={Production Development Test}]	<p>Optional. The type of installation for which to retrieve license key information. Values are:</p> <ul style="list-style-type: none"> <li data-bbox="732 953 922 980">■ Production</li> <li data-bbox="732 1012 938 1039">■ Development</li> <li data-bbox="732 1071 834 1098">■ Test</li> </ul>
[ <code>--input   -i</code> ] [ <i>filename</i> {.xml .json}]	<p>Optional. Identifies an input file that contains the data for the new node. For more information, see <a href="#">“input” on page 204</a>.</p> <div data-bbox="732 1251 1459 1472" style="background-color: #f0f0f0; padding: 10px;"> <p><b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get landscape nodes</a> to retrieve data for an existing node. For example, if you want to use an XML input file, use <a href="#">sagcc get landscape nodes</a> with the <code>--format xml</code> option to retrieve the data in XML format.</p> </div>
[ <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 <a href="#">“Common Options” on page 193</a>.</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 215](#) and [“username” on page 219](#). 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
[{alias   nodeAlias=alias1 [nodeAlias=alias2 ... nodeAlias=aliasn]]	<p>Optional. Specifies the alias name(s) of the installation(s) you want to remove.</p> <p>If you execute the <code>sagcc delete landscape nodes</code> command without supplying alias names, the command removes all installations it is currently managing.</p> <p>To remove a single installation, supply its alias name using the following format:</p> <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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#), [“username” on page 219](#), and [“password” on page 212](#).

- 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>	Required. Specifies the alias name of the installation for which you want to generate an ID.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
<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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#).

## 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 <a href="#">sagcc list landscape nodes</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

### 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 <a href="#">“Common Options” on page 193</a> .

## 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] [installationType={Production|Development|Test}] [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 <a href="#">sagcc list landscape nodes</a>.</p>
[ <i>name=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>
[ <i>description=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>
[ <i>installationType</i> ={ <i>Production</i>   <i>Development</i>   <i>Test</i> }]	<p>Optional. The type of installation for which to retrieve license key information. Values are:</p> <ul style="list-style-type: none"> <li>■ Production</li> <li>■ Development</li> <li>■ Test</li> </ul>

Argument or Option	Description
<code>[{--input   -i} filename{.xml .json}]</code>	Optional. Identifies an input file that contains the updated data for the landscape. For more information, see <a href="#">“input” on page 204</a> .  <b>Tip:</b> To determine how to specify the data in the input file, use the <a href="#">sagcc get landscape nodes</a> to retrieve data for the node you want to update. For example, if you want to use an XML input file, use <a href="#">sagcc get landscape nodes</a> 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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). The command specifies “secret” for the user’s password.

## License Keys Commands

### 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> ]	Optional. 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.  If you do not specify this argument, Command Central generates the license key alias automatically.
{--input -i} <i>filename</i>	Required. Specifies the name of the input file. For more information, see <a href="#">“input” on page 204</a> .
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

The license key manager supports the following 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
```

- To add the license key file "0000028138\_WOK\_9.12\_PROD\_LNXAMD64.xml" without specifying a license key alias:

```
sagcc add license-tools keys
-i C:\Licenses\0000028138_WOK_9.12_PROD_LNXAMD64.xml
```

## sagcc get license-tools keys

Retrieves license key files from the Command Central license key manager.

### Syntax

- Command Central syntax:

- To retrieve all license key files and download them in a zip archive file:

```
sagcc get license-tools keys [-o|--output] filename.zip
[options]
```

- To retrieve and download a license key file with the specified alias:

```
sagcc get license-tools keys licenseKeyAlias
[{-o|--output} filename] [options]
```

- Not supported on Platform Manager.

### Arguments and Options

The following table lists the command arguments for downloading all license files.

Argument or Option	Description
<code>-o --output filename</code>	Required. Specifies the name of the output zip archive file. For more information, see <a href="#">“output” on page 209</a> .
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

The following table lists the command arguments for downloading a license file by license alias.

Argument or Option	Description
<code>licenseKeyAlias</code>	Required. The alias of the license key file to retrieve.
<code>[-o --output filename]</code>	Optional. Specifies the name of the output file. For more information, see <a href="#">“output” on page 209</a> .
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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.
- After you download all license keys from a Command Central server as a zip archive, you can import the zip archive into another Command Central server, using the [“sagcc add license-tools keys” on page 325](#) command.

## Examples When Executing on Command Central

- To retrieve all license key files from the license key manager for the "ccprod" server and download them in a zip archive file named "CC-license-keys.zip":

```
sagcc get license-tools keys -o c:\temp\CC-license-keys.zip -s http://ccprod:8090/cce
```

You can then add the downloaded "CC-license-keys.zip" into the Command Central server with name "ccdev":

```
sagcc add license-tools keys -i CC-license-keys.zip -s http://ccdev:8090/cce
```

- To retrieve the 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|current]
[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.
[productId=productId]	Optional. The ID of the product for which to retrieve license key information.
[os=operatingSystemCode1,operatingSystemCode2...operatingSystemCodeN]	Optional. The operating system or systems for which to retrieve license key information.
[release=releaseVersion current]	Optional. The release version of the product for which to retrieve license keys. For the current release version specify <code>release=current</code>
[platform=platformCode1,platformCode2...platformCodeN]	Optional. The operating system platform or platforms for which to retrieve license key information.
[installationType={Production Development Test}]	Optional. The type of installation for which to retrieve license key information. Values are: <ul style="list-style-type: none"> <li>■ Production</li> <li>■ Development</li> <li>■ Test</li> </ul>
[licenseKeyType={Standard Custom}]	Optional. The type of license key for which to retrieve information. Values are: <ul style="list-style-type: none"> <li>■ standard</li> <li>■ custom</li> </ul>

Argument or Option	Description
[ <i>aliasFilter=filter</i> ]	Optional. A search string that filters the license keys by aliases matching the string.
[ <i>excludeExpired=true false</i> ]	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>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- The license key manager supports the following types of license keys:
  - **Standard.** Single 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.
- The command returns a list of all license keys registered in Command Central, including license keys added individually and license keys added with a license manifest file.
- The details of a license key in the command output show whether a license key is added by a user or from a license manifest file. If a license is added from a license manifest file, the Added By column for the license key has the alias name of the manifest file, followed by (M).

## 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>	Optional. The existing license key alias to which to assign a new license key file.  If you do not specify this argument, Command Central generates the license key alias automatically.
<i>{--input -i} filename</i>	Required. Specifies the name of the input file. For more information, see <a href="#">“input” on page 204</a> .
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

### Usage Notes

- The license key manager supports the following 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 cannot delete embedded license keys with this command. To delete license keys that were added to Command Central with a license manifest file, you must delete the entire license manifest file using the `“sagcc delete license-tools manifests”` on page 343 command.

## 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.

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 <a href="#">“Common Options” on page 193</a> .

---

## 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 325](#).

## 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
```

## License Reports Commands

---

### sagcc create license-tools reports installation

Creates an installation report for the current landscape. For more information about installation reports, see [“Monitor the Licensing State for a Landscape” on page 46](#).

#### Syntax

- Command Central syntax:

```
sagcc create license-tools reports installation [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 193.

## Usage Notes

To check the status of the job scheduled to create the installation 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 installation report 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 installation 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 193.

## Example When Executing on Command Central

To delete an installation report with report ID c4eff7:

```
sagcc delete license-tools reports installation c4eff7
```

## sagcc delete license-tools reports installation

Deletes all generated installation reports 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
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see “Common Options” on page 193.

## sagcc get license-tools reports installation output PDF

Generates a PDF file for an existing installation report.

### 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
reportid	Required. The ID of the installation report for which to generate a PDF file.
--output-format pdf	Required. Specifies that the output is generated in PDF format.
--output filename.pdf	Required. Specifies a name for the output PDF file. For more information, see “output” on page 209.

## 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 installation report output data should be generated in PDF format.

## Example When Executing on Command Central

To generate a PDF file, named `report.pdf`, for an installation 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 installation report 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 installation 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-<i>filename.xml</i></code>	Required. Specifies a name for the output XML file. For more information, see “output” on page 209.

## Example When Executing on Command Central

To generate an XML file, named `report.xml`, for an installation 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 list license-tools reports installation

Lists all installation reports 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 193.

---

### 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 installation-report

Deprecated without replacement. Lists all installation reports available on the Command Central server.

### Syntax

- Command Central syntax:

```
sagcc list license-tools reports installation-report [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 193.

## 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 create license-tools reports installation-report

Deprecated without replacement. Creates an installation report based on the currently registered nodes in a Command Central instance.

### Syntax

- Command Central syntax:

```
sagcc create license-tools reports installation-report [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 193.

## Usage Notes

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

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

## sagcc get license-tools reports installation-report reportid

Deprecated without replacement. Obtains information about an installation report with the specified unique report identifier.

### Syntax

- Command Central syntax:

```
sagcc get license-tools reports installation-report 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 193.

### Examples When Executing on Command Central

To obtain information about a report with report ID c4eff6:

```
sagcc get license-tools reports installation-report c4eff6
```

## sagcc delete license-tools reports installation-report

Deprecated without replacement. Deletes the specified installation report from Command Central.

### Syntax

- Command Central syntax:

```
sagcc delete license-tools reports installation-report [reportid] [options]
```

```
options:  
[--force]
```

- Not supported on Platform Manager.

## Arguments and Options

Argument or Option	Description
[ <i>reportid</i> ]	Optional. The ID of the report, which delete. If you omit this argument, the command deletes all installation reports from Command Central.
[ <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 193.

## sagcc list license-tools reports

Lists all license reports, by license key and by license manifest file, available on the Command Central server.

### Note:

License metering and license manifest files are deprecated without replacement in Command Central

## Syntax

- Command Central syntax:

```
sagcc list license-tools reports [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 193.

## 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

Deprecated without replacement. 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
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see “Common Options” on page 193.

---

## sagcc add license-tools manifests

Deprecated without replacement. 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
manifestAlias	Required. The alias for the license manifest file to add.

---

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 204.
<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 193.

## Usage Notes

If a license manifest file contains embedded license keys, the license keys are added to Command Central when you add the license manifest. You can view all license keys registered in Command Central with the “[sagcc list license-tools keys](#)” on page 328 command.

## 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

Deprecated without replacement. 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.
<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 193.

## 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

Deprecated without replacement. 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 209.
[ <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 193.

### 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

Deprecated without replacement. 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
[ <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 193.

## sagcc list license-tools manifests content

Deprecated without replacement. 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 193.

## 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	LNXAMD64
HP-UX	V7	HP11
MAC OS	V8	OSX
OS/400 RISC	W9	AS400
Oracle Linux	YO	LNXAMD64
Red Hat Linux	YC	LNXAMD64
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 get license-tools metering contracts licensekey

Deprecated without replacement. 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
[ <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 automatic 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 <a href="#">“Common Options” on page 193</a> .

## 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 list license-tools manifests contracts

Deprecated without replacement. 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 “sagcc list inventory products” on page 296 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 193.

### 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
```

## Lifecycle Commands

### 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"> <li>■ <i>start</i> - starts the run-time component</li> <li>■ <i>startindebugmode</i> - starts the run-time component in debug mode</li> <li>■ <i>startinsafemode</i> - starts the run-time component in safe mode</li> <li>■ <i>stop</i> - stops the run-time component</li> <li>■ <i>restart</i> - stops, then restarts the run-time component</li> <li>■ <i>pause</i> - pauses the run-time component</li> <li>■ <i>resume</i> - resumes previously paused run-time component</li> </ul> <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>
<i>node_alias</i>	Command Central only.

Argument or Option	Description
	Required when you do not specify search criteria. Specifies the alias name of an installation. You can determine installation alias names using the <a href="#">sagcc list landscape nodes</a> command.
<i>componentid</i>	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 <a href="#">sagcc list inventory components</a> command.  <b>Note:</b> Command Central only requires the <i>componentid</i> when you do not specify search criteria.  When executing the command against a Platform Manager, specify the <i>componentid</i> before the action in the command syntax.
[ <i>criteria</i> ]	Command Central only.  Optional. Specifies to act only on the run-time components that match the search criteria you specify. For more information, see <a href="#">“Specifying Search Criteria for Lifecycle Commands” on page 351</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## 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 352](#).
- You cannot stop the OSGI-SPM component using the `sagcc exec 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 215](#), [“username” on page 219](#), and [“password” on page 212](#).

- 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 215](#) and [“username” on page 219](#). 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 219](#). 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
- 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
```

## Monitoring Commands

---

### 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
{ <i>runtimestatus</i>   <i>runtimestate</i>   <i>state</i> }	Required. Specifies whether you want to retrieve run-time statuses, run-time states, or state.
<i>node_alias</i>	Command Central only.  Required. Specifies the alias name of the installation on which the run-time component is installed.

Argument or Option	Description
	You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
<i>componentid</i>	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 <a href="#">sagcc list inventory components</a> .
[ <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 <a href="#">“Common Options” on page 193</a> .

## 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 215](#) and [“username” on page 219](#). 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 215](#) and [“username” on page 219](#). 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 219](#). 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 <a href="#">sagcc list landscape nodes</a>.</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 <a href="#">sagcc list inventory components</a>.</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"> <li>■ On Command Central:           <p>When you include this argument, Command Central retrieves monitoring data from the Platform Manager node. When you do not include this argument, Command Central retrieves monitoring data from its cache.</p> </li> <li>■ On Platform Manager:           <p>When you include this argument, Platform Manager requests the current monitoring state directly from the product.</p> </li> </ul>
[options]	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</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 <a href="#">sagcc list landscape nodes</a> .
[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 <a href="#">sagcc list inventory components</a> .
[includeChildren=true]	Optional. Indicates whether to retrieve the status and state for a run-time component and its dependent components.  When you include this argument, Command Central retrieves monitoring data for the run-time component and all of its dependent components.  When you do not include this argument, Command Central retrieves monitoring data <i>only</i> for the run-time component.

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 <a href="#">“Common Options” on page 193</a> .

## 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 219](#). The command specifies “secret” for the user’s password.

## Provisioning Bootstrap Installers Commands

### sagcc list provisioning bootstrap installers

Retrieves a list of bootstrap installers available in Command Central.

To add bootstrap installers in Command Central, download the Command Central bootstrap installer for the required release and operating system from the Empower Product Support website and save the downloaded installer file into the \$CC\_HOME/profiles/CCE/data/installers directory.

### Syntax

- Command Central syntax:

```
sagcc list provisioning bootstrap installers [version=version]
[platform=platformCode][options]
```

- Not supported on Platform Manager.

### Options

Option	Description
<i>installerName</i> {.sh   .bat}	The name of the installer file to find (for example, cc-def-10.7-lnxadm64.sh).
<i>version=version</i>	The version of the bootstrap installer to find.

Option	Description
[platform= <i>platformCode</i> ]	Code of the target operating system of the bootstrap installer to find.
[ <i>options</i> ]	Any options supported by the command line interface (see <a href="#">“Common Options” on page 193</a> ).

## Examples When Executing on Command Central

- To list all available bootstrap installers:

```
sagcc list provisioning bootstrap installers
```

- To list bootstrap installers for repository version “10.11” and operating system “OSX”:

```
sagcc list provisioning bootstrap installers version=10.11
platform=OSX
```

## sagcc exec provisioning bootstrap nodes

Installs a Platform Manager node on a local or remote machine using the Command Central bootstrap installer, specified in the input file, and registers the new Platform Manager installation under a specified node alias.

### Syntax

- Command Central syntax:

```
sagcc exec provisioning bootstrap nodes nodeAlias [isSecure={true | false}]
[credentialsAlias=credentialsAlias]
-i bootstrap-info-file [options]
```

- Not supported on Platform Manager.

### Options

Option	Description
<i>nodeAlias</i>	Required. Alias name of the bootstrapped Platform Manager installation that you want to add to the target machine.
[isSecure={true   false}]	Optional. Indicates whether to use HTTPs to connect to Platform Manager. Valid values: <ul style="list-style-type: none"> <li>■ true - Use HTTPs.</li> <li>■ false (default) - Do not use HTTPs.</li> </ul>

Option	Description
[credentialsAlias=credentialsAlias]	Required when using Command Central 10.11 or higher. The credentials alias to use for the Administrator user for Platform Manager. See the usage notes for more details about this argument.
-i bootstrap-info-file	Required. An input file in XML format that specifies the bootstrap details for installing Platform Manager. For examples of bootstrap info files in XML format, see the "Example Bootstrap Info Files".
[options]	Optional. Any common option supported by the Command Line Interface (see <a href="#">"Common Options" on page 193</a> )

## 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.
- You can download the Command Central bootstrap installer for the release that you require 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 specify the installer in the input file when running this command.
- Based on the release version of Command Central, when you include the `[credentialsAlias=credentialsAlias]` argument, the command does the following:
  - With Command Central 10.11 or higher, the value for the `[credentialsAlias=credentialsAlias]` argument is required for any Platform Manager version that you want to install. Software AG strongly recommends that you use a credentials alias with a strong password. You can set the strong password either in the ADMINISTRATOR credentials alias (which does not have a default password value), or create a custom credentials alias.
  - With Command Central 10.7 or lower, the value for the `[credentialsAlias=credentialsAlias]` argument is not required for any Platform Manager version that you want to install. If you omit the argument, the command uses the DEFAULT\_ADMINISTRATOR credentials alias, which has a default password value. Software AG strongly recommends that you use a custom credentials alias with a strong password.

For information about adding credentials aliases, see ["Create Credentials Aliases" on page 16](#).

- 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 200](#), [“wait” on page 220](#), and [“check-every” on page 196](#).

- The input file that specifies the bootstrap details for installing Platform Manager uses the following XML schema:

- To install Platform Manager on the local machine:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bootstrapInfo>
  <hostname>localhost</hostname>
  <installer>bootstrapInstaller</installer>
  <installDir>installationDirectory</installDir>
  <httpPort>portNumber</httpPort>
  <httpsPort>portNumber</httpsPort>
</bootstrapInfo>
```

- To install Platform Manager on a remote machine:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bootstrapInfo>
  <hostname>remoteMachine</hostname>
  <installer>bootstrapInstaller</installer>
  <port>portNumber</port>
  <credentials>
    <authenticationType>BASIC</authenticationType>
    <authenticationMethod>PASSWORD|INTERACTIVE</authenticationMethod>
    <userName>user</userName>
    <password>password</password>
  </credentials>
  <substituteUserCredentials>
    <authenticationType>BASIC</authenticationType>
    <authenticationMethod>PASSWORD</authenticationMethod>
    <userName>user</userName>
    <password>password</password>
  </substituteUserCredentials>
  <installDir>/home/localUser/softwareag</installDir>
  <httpPort>portNumber</httpPort>
  <httpsPort>portNumber</httpsPort>
</bootstrapInfo>
```

The following table describes the parameters in the input file:

hostname	Required. The hostname of the machine on which to install Platform Manager.
installer	Required. The bootstrap installer file to use when installing Platform Manager.
port	Required when bootstrapping on a remote machine. The SSH port number to use to connect to the remote machine.

credentials	Required when bootstrapping on a remote machine. Includes information about the SSH credentials to use to connect to the remote machine.
authenticationType	Required when bootstrapping on a remote machine. Specifies the authentication type to use when connecting to the remote machine. Valid values: <ul style="list-style-type: none"> <li>■ NONE</li> <li>■ BASIC</li> <li>■ TRUSTED</li> </ul>
authenticationMethod	Required when bootstrapping on a remote machine. Specifies the authentication method to use when connecting to the remote machine. Valid values: <ul style="list-style-type: none"> <li>■ PASSWORD</li> <li>■ INTERACTIVE</li> <li>■ CERTIFICATE</li> </ul>
userName	Required for authenticationType: BASIC and authenticationMethod: PASSWORD. Specifies the SSH user name.
password	Required for authenticationType: BASIC and authenticationMethod: PASSWORD. Specifies the SSH user password.
substituteUserCredentials	Optional. Specifies the details of a substitute user to use to install and start Platform Manager.
installDir	Required. The directory in which to install Platform Manager.
httpPort	Required. The Platform Manager HTTP port.
httpsPort	Required. The Platform Manager HTTPS port.

### Example Bootstrap Info Files

To install Platform Manager on the local machine in the "C:\sag\cc" installation directory, using the "cc-def-10.11-fix1-w64.bat" bootstrap installer and the "9092" (HTTP) and "9093" (HTTPS) ports:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bootstrapInfo>
  <hostname>localhost</hostname>
  <installer>cc-def-10.11-fix1-w64.bat</installer>
  <installDir>C:\sag\cc</installDir>
```

```
<httpPort>9092</httpPort>
<httpsPort>9093</httpsPort>
</bootstrapInfo>
```

To install Platform Manager in the "/home/localUser/softwareag installation directory of a remote machine with hostname "sag1", using the "cc-def-10.7-fix5-lnxamd64.sh" bootstrap installer and the "9092" (HTTP) and "9093" (HTTPS) ports. The SSH user to connect to the remote machine is "sagSshUser" and "localUser" is the substitute user that will install and start Platform Manager:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bootstrapInfo>
  <hostname>sag1</hostname>
  <installer>cc-def-10.7-fix5-lnxamd64.sh</installer>
  <port>22</port>
  <credentials>
    <authenticationType>BASIC</authenticationType>
    <authenticationMethod>PASSWORD</authenticationMethod>
    <userName>sagSshUser</userName>
    <password>encryptedValue</password>
  </credentials>
  <substituteUserCredentials>
    <authenticationType>BASIC</authenticationType>
    <authenticationMethod>PASSWORD</authenticationMethod>
    <userName>localUser</userName>
    <password>encryptedValue</password>
  </substituteUserCredentials>
  <installDir>/home/localUser/softwareag</installDir>
  <httpPort>9092</httpPort>
  <httpsPort>9093</httpsPort>
</bootstrapInfo>
```

## Example When Executing on Command Central

To install Platform Manager on the local machine using the bootstrap details provided in the bootstrapInfo.xml file, the "myNodeCredentials" credentials alias for the Platform Manager Administrator user, and HTTPS as the transport protocol:

```
sagcc exec provisioning bootstrap nodes localhost isSecure=true
credentialsAlias=myNodeCredentials
-i bootstrapInfo.xml
```

## Provisioning Fixes Commands

### 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 35](#).

## Syntax

- Command Central syntax:

```
sagcc exec provisioning fixes node_alias repo_name install
[products=productId,productId]
[artifacts=fixName1[_version],fixName2[_version] |
FixId1, FixId2] [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>productId1</i> , <i>productId2</i> ]	Optional. The IDs of the products for which you want to install fixes.
[artifacts= <i>fixName1[_version]</i> , <i>fixName2[_version]</i>   <i>FixId1</i> , <i>FixId2</i> ]	Optional. The name or fix ID of the fix or fixes to install.  If you specify fix names, you can specify the fix version.  The fix ID is the official ID of the fix on Empower.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 fix repository. If you specify a fix name without a version in the `artifacts` argument, the command installs the latest fix with the specified name. If you specify fix names with a version, or a fix ID in the `artifacts` argument, the command installs only the specified fixes. The difference between using a fix name with a version and the Empower fix ID is the version format. If the list of fix names or IDs includes engineering fixes, the command installs the specified engineering fixes.

To find the fix names or fix IDs, run the `sagcc list repository fixes content` command or view the contents of the fix repository in the Command Central web user interface.

- 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
```

- To install the fixes with IDs “CCE\_10.2\_CCE\_SPMplugin\_Fix2” and “CCE\_10.2\_CLI\_Fix2” 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=CCE_10.2_CCE_SPMplugin_Fix2,CCE_10.2_CLI_Fix2
```

- To install the fix with ID “CCE\_10.2\_CCE\_SPMplugin\_Fix2” and the latest fix with name “wMFix.SPM.TEST1” 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=CCE_10.2_CCE_SPMplugin_Fix2,wMFix.SPM.TEST1
```

## sagcc exec provisioning fixes uninstall

Uninstall a fix. The uninstall operation does not require access to a repository.

### Syntax

- Command Central syntax:

```
sagcc exec provisioning fixes nodeAlias uninstall
artifacts={fixName1,fixName2 | ALL} [options]
```

- Not supported on Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the installation from which to uninstall fixes.
<i>artifacts</i> ={ <i>fixName1</i> , <i>fixName2</i>   ALL}	Required. The fix names of the fixes to uninstall. To uninstall all fixes, set <i>artifacts</i> =ALL.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see “ <a href="#">Common Options</a> ” on page 193.

## Usage Notes

- When uninstalling a fix, Platform Manager restores the previously installed version of the fix. For example, when you uninstall SPM\_10.3\_Fix2, the command rolls back SPM\_10.3\_Fix1.
- Before you run the fixes uninstall command, check which fixes depend on the fix you intend to uninstall. Command Central uninstalls the fix and all fixes that depend on the uninstalled fix. Use the `sagcc list provisioning fix uninstall dependencies` command to check which fixes depend on the fix you are going to uninstall.
- To find the fix names, run the `sagcc list repository fixes content` command or view the contents of the fix repository in the Command Central web user interface.

## Examples When Executing on Command Central

- To uninstall a fix with name “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 artifacts=ALL
```

## sagcc list provisioning fixes uninstall dependencies

Returns a list of all fixes that depend on the specified fix(es).

### Syntax

- Command Central syntax:

```
sagcc list provisioning fixes nodeAlias uninstall dependencies  
[artifacts=Id1,Id2]
```

- Not supported on Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the installation in which the specified fixes are installed.
[ <i>artifacts=Id1,Id2</i> ]	Optional. The IDs of the fixes for which to list dependent fixes.  <b>Tip:</b> Specify at least one fix ID in the artifacts argument, otherwise Command Central lists all fixes installed on the specified node.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Examples When Executing on Command Central

To list all fixes that depend on the fix with ID “wMFix.SPM.TEST” installed on the node with alias “sag01”:

```
sagcc list provisioning fixes sag01 uninstall dependencies artifacts=wMFix.SPM.TEST
```

## Provisioning Products Commands

### 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 33](#).

### Syntax

- Command Central syntax:

```
sagcc exec provisioning products node_alias repo_name install  
[artifacts=productId[_version],productId[_version]]  
[administratorCredentials=credentialsAlias]  
[administratorCredentialsChangeOnFirstLogin={true|false}][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.
[ <i>artifacts=productId[_version],productId[_version]</i> ]	<p>The names of the products to install.</p> <ul style="list-style-type: none"> <li>■ Required for a master repository. You must specify at least one product with version.</li> <li>■ Optional for image repositories. If you do not specify any artifacts, Command Central installs all products in the image archive.</li> </ul> <p>When you specify more than one product, specifying the version is optional.</p> <div style="background-color: #f0f0f0; padding: 5px;"> <p><b>Note:</b> To find the product IDs, use the <a href="#">sagcc list repository products content</a> command.</p> </div>
[ <i>administratorCredentials=credentialsAlias</i> ]	Optional. Use only when installing products of version 10.11 or higher. The credentials alias to use for the product administrator user. If you omit this argument, the command uses the ADMINISTRATOR credentials alias. See the usage notes for more details about this argument.
[ <i>administratorCredentialsChangeOnFirstLogin={true false}</i> ]	<p>Optional. Use only when installing products in installations of version 10.11 or higher. Indicates whether the product requires changing the password of the product administrator user on the initial login. Valid values:</p> <ul style="list-style-type: none"> <li>■ true - password change required.</li> <li>■ false (default) - no password change required.</li> </ul>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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.
- When listing a product in the `artifacts` argument, you must enclose the `product Id` in single quotes (") if the ID includes underscore (\_), followed by a number. The single quotes ensure that Command Central does not interpret the (\_) character in the `product Id` as the version delimiter in `[artifacts=productId[_version]]`. For example, to list the product with ID "WmPSFT\_E1\_898" and version 8.98.6282010.0.0, enter `artifacts='WmPSFT_E1_898'` (without the optional version) or `artifacts='WmPSFT_E1_898' _8.98.6282010.0.0` (with the version).
- When installing products with version 10.11 and higher, by default the command uses the ADMINISTRATOR credentials alias. Note that the ADMINISTRATOR credentials alias does not have a default password value and you must specify a strong administrator password by navigating to **Environments > ALL > Instances > CCE > Command Central Server > Configuration > Credentials** in the Command Central web user interface and editing the ADMINISTRATOR credentials. Optionally, you can use `[administratorCredentials=credentialsAlias]` to specify a custom credentials alias that you defined for the product administrator user. For information about adding credentials aliases, see [“Create Credentials Aliases” on page 16](#).

**Important:**

If you specify a custom credentials alias, Command Central will set the provided password for the default administrator user of the product and ignore the username value of the credentials alias. For example, if you specify a credentials alias that has a username value "my-user" and a password value "my-admin-password" and the default administrator user for the product you are installing is "sysadmin", then Command Central will set the password "my-admin-password" to the "sysadmin" user when installing the product.

- When installing products with version 10.7 and lower, the command does not provide credentials for the product administrator user. Command Central installs each product with the default administrator user and password for the product.

**Examples When Executing on Command Central**

- To install all products with version 10.11 from a mirror repository with name "MyProducts-1011" on a target node with alias "sag01" using the ADMINISTRATOR credentials alias:

```
sagcc exec provisioning products sag01 MyProducts-1011 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_10.7,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 nodeAlias uninstall
artifacts=Id1,Id2 [options]
```

- Not supported on Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the installation from which to uninstall products.
<i>artifacts=Id1,Id2</i>	Required. The IDs of the products that you want to uninstall.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Usage Notes

- Before you run the products uninstall command, check which products depend on the product you intend to uninstall. Command Central uninstalls the product and all run-time components that depend on the uninstalled product. For example, Platform Manager depends on a run-time component. Uninstalling the dependent run-time component will also uninstall Platform Manager. Use the `sagcc list provisioning products uninstall dependencies` command to check which products depend on the product you are going to uninstall.

### 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
```

## sagcc list provisioning products uninstall dependencies

Returns a list of all products that depend on the specified product(s).

### Syntax

- Command Central syntax:

```
sagcc list provisioning products nodeAlias uninstall dependencies
[artifacts=Id1,Id2]
```

- Not supported on Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>nodeAlias</i>	Required. The alias of the installation in which the specified products are installed.
[artifacts= <i>Id1,Id2</i> ]	Optional. The IDs of the products for which to list dependent products.  <b>Tip:</b> Specify at least one product ID in the artifacts argument, otherwise Command Central lists all products installed on the specified node.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Examples When Executing on Command Central

To list all products that depend on the run-time components with IDs “integrationServer” and “ISspm” installed on the node with alias “sag01”:

```
sagcc list provisioning products sag01 uninstall dependencies
artifacts=integrationServer,ISspm
```

## Repository Commands

### About Repository Commands

The following table describes the type of source repositories that you can add and manage with the repository commands:

Source repository type	Use to...
Product	Install Software AG products
Fix	Install fixes for the products managed by Command Central.
Asset	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> .

## Using HTTPS to Connect to Repositories

Command Central can register a repository on the HTTP, HTTPS, or both ports based on the following:

- The version of the Platform Manager installation
- The repository type
- Whether the HTTP and HTTPS ports are enabled. If a port is disabled, Command Central cannot register a repository on that port.

When the HTTP and HTTPS ports are enabled in a Platform Manager installation version 10.5 or higher, Command Central registers a repository hosted in that installation at both locations (<http://> and <https://>). You can check the available locations for a repository using the “[sagcc list repository](#)” on page 400 command.

### Note:

Software AG recommends that you do not disable the HTTP port when provisioning product installations with version 10.1 or lower. You cannot provision products with version 10.1 or lower using a mirror repository if the HTTP port is disabled.

You must consider the following when you want to connect to a repository registered in Command Central over HTTPS.

- When installing products or fixes from a master repository, Command Central uses HTTPS to connect to 9.12 or higher repositories and HTTP to connect to 9.10 or lower repositories.
- When installing products or fixes hosted on a local or remote Platform Manager with version 10.5 or higher from a mirror repository, Command Central uses HTTPS to connect to 10.3 or higher repositories and HTTP to connect to 10.1 or lower repositories.

## 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
<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>	The path to the directory on the Command Central server that contains the flat file repository or the zip archive of the flat file.
<i>-i flat-file-repo-zip-on-client</i>	The path to the zip archive of the flat file repository on a client machine.
[ <i>registryRoot=URL</i> ]	Optional. A valid URL that points to the location of the registry that stores the metadata for the assets in the flat file 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:\fileRepo directory and the registry directory is "MyRegistry", you must set this parameter as follows: <code>registryRoot=/MyRegistry</code>
[ <i>registryNamespacesRoot=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: <code>registryNamespacesRoot=/MyRegistryNs</code>
[ <i>description=description</i> ]	Optional. A description of the asset repository.
[ <i>overwrite=true false</i> ]	Optional. Use this argument when re-posting build artifacts from a Jenkins CI job. When the argument is set to true: <ul style="list-style-type: none"> <li>■ If the repository does not exist, Command Central creates the repository.</li> </ul>

Argument or Option	Description
	<ul style="list-style-type: none"> <li>■ If the repository exists, Command Central removes the existing repository and creates a new one.</li> </ul> <p>The default is <code>true</code>.</p>
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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](#) 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
<i>name=repoName</i>	Required. A unique name for the remote Git repository that you are adding.
<i>location=URL</i>	Required. A valid URL that points to the location of the remote Git repository. For example, <a href="https://github.com/user/repo.git">https://github.com/user/repo.git</a>
[ <i>registryRoot=URL</i> ]	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 C:\fileRepo directory and the registry directory is "MyRegistry", you must set this parameter as follows: registryRoot=/MyRegistry
[ <i>registryNamespacesRoot=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

Argument or Option	Description
<code>credentials=credAlias</code>	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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with the “ <a href="#">sagcc create configuration data</a> ” on page 233 command. See the “ <a href="#">Usage Notes</a> ” on page 377 for details about the GIT credentials input properties file that you must provide with the <code>sagcc create configuration data</code> command.
<code>[overwrite=true false]</code>	Optional. Use this argument when re-posting build artifacts from a Jenkins CI job. When the argument is set to <code>true</code> : <ul style="list-style-type: none"> <li>■ If the repository does not exist, Command Central creates the repository.</li> <li>■ If the repository exists, Command Central removes the existing repository and creates a new one.</li> </ul> <p>The default is <code>true</code>.</p>
<code>[description=description]</code>	Optional. A description of the asset repository.
<code>[branch=branch_name]</code>	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. <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p><b>Note:</b> Use a simple branch name. Do not use prefixes, such as <code>origin/*</code> or <code>refs/*</code></p> </div>
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see “ <a href="#">Common Options</a> ” on page 193.

## 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 233 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 488.

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](#) 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 <a href="#">“Common Options” on page 193</a> .

## 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
- Type
- Location
- Time of last update
- Repository description, or null if none is assigned

## Syntax

- Command Central syntax:

```
sagcc list repository assets [repoName] [options]
```

- Not supported by Platform Manager.

## Arguments and Options

Argument or Option	Description
[ <i>repoName</i> ]	Optional. The name of the asset repository for which to list information. If you omit this argument, Command Central lists information about all registered asset repositories.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- To get the repository description in the output, include `properties=*` in the command. For details about how to use this argument, see [“properties” on page 213](#).
- 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
```

To list the details and the description of the repository with name "MyAssetRepo", registered on the local Command Central server:

```
sagcc list repository assets MyAssetRepo properties=*
```

Name	Type	Location	Created By	Description	Branch
MyAssetRepo	FLATFILE	file:/c:/sag/cc/profiles/CCE/data/flatfile/MyAssetRepo.zip	Administrator	templates assets repository	
Last Updated		Thu May 28 11:23:34 EEST 2020			

## 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.
[type=composite asset]	<p>Optional. Indicates which type of metadata to retrieve.</p> <ul style="list-style-type: none"> <li>■ type=composite - list metadata for composite assets</li> <li>■ type=asset - list metadata only for assets</li> </ul> <p>When type is not included, the default is type=composite</p>
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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.
<code>type=asset</code>	Required. Indicates that you want to retrieve individual assets from the available composites.
<code>asset=assetName</code>	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 <a href="#">“Common Options” on page 193</a> .

## 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.

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 <a href="#">“Common Options”</a> on page 193.

## 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]
[credentials=credAlias] [registryRoot=URL] [registryNamespacesRoot=URL]
[description=description] [options]
```

- Not supported by Platform Manager.

### Arguments and Options

Argument or Option	Description
name= <i>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.
[-i <i>flat-file-repo-zip-on-client</i> ]	Optional. The path to the zip archive of the flat file repository on a client machine.
[ <i>credentials=credAlias</i> ]	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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with the <a href="#">“sagcc create configuration data”</a> on page 233 command. See the <a href="#">“Usage Notes”</a> on page 377 for details

Argument or Option	Description
	about the GIT credentials input properties file that you must provide with the <code>sagcc create</code> configuration data command.
<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 <code>C:\fileRepo</code> 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 <code>C:\fileRepo</code>, 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 <a href="#">“Common Options” on page 193</a> .

## 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](#) 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, <code>https://github.com/user/repo.git</code>
<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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with the “ <a href="#">sagcc create configuration data</a> ” on page 233 command. See the “ <a href="#">Usage Notes</a> ” on page 377 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 <code>C:\fileRepo</code> directory and the registry directory is "MyRegistry", you must set this parameter as follows: <code>registryRoot=/MyRegistry</code>
<code>[registryNamespacesRoot=<i>URL</i>]</code>	Optional. A valid URL that points to the location of the directory that stores the registry namespaces.

Argument or Option	Description
	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.  <b>Note:</b> 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 <a href="#">“Common Options” on page 193</a> .

## 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](#) 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=<i>repo_name</i></code>	Required. The name of the fix repository to add.
<code>[description="<i>description</i>"]</code>	Optional. A description for the repository.
For master repository	
<code>location=<i>remote_location</i></code>	Required. The URL of the remote master repository.
<code>credentials=<i>credAlias</i></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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with <a href="#">"sagcc create configuration data"</a> on <a href="#">page 233</a> .
For mirror repository	
<code>sourceRepos=<i>repo_name1,repo_name2...repo_nameN</i></code>	Required. A list of the source repositories to include in the new mirror repository.

Argument or Option	Description
<code>productRepos=repo_name1,repo_name2...repo_nameN</code>	Required. Select product repositories for which to add fixes in the fix mirror repository. See the <a href="#">“Usage Notes” on page 388</a> 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 <a href="#">“input” on page 204</a> .
<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 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.
	<b>Important:</b> Two repositories cannot point to the same location.

## 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=<i>repo_name</i></code>	Required. The name of the product repository to add.  The name of a mirror repository should follow this format: <code>webMethods-<i>major.minor_label</i></code>  The <i>label</i> could be location, platform, or anything meaningful to identify the mirror repository. For example, <code>webMethods-10.5_US_east_linux</code>
<code>[description="<i>description</i>"]</code>	Optional. A description for the repository.
For master repository	
<code>location=<i>remote_location</i></code>	Required. The URL of the remote master repository. To find the URL of the available Software AG product repositories use <a href="#">“sagcc list repository discover” on page 401</a> .

Argument or Option	Description
<code>credentials=credAlias</code>	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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with <a href="#">“sagcc create configuration data” on page 233</a> .
<b>For mirror repository</b>	
<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>sourceRepos=repo_name1,repo_name2...repo_nameN</code>	Required. A list of the source repositories to include in the new mirror repository.
<code>[artifacts=productId1,productId2...productIdN]</code>	Optional. A list of the IDs of the products from the source repositories that you want to include in the mirror repository.  If you do not specify a list of product IDs or you omit this parameter, Command Central includes all products from the source repositories.
<code>[platforms=OS_Id1,OS_Id2...OS_IdN]</code>	Optional. A list of the IDs of the operating systems of the products included in the mirror repository.  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.
<b>For image repository</b>	
<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 <a href="#">“input” on page 204</a> .
<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 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><b>Important:</b> Two repositories cannot 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.
- You can create a product mirror repository from one or more product image repositories. When you use a product image repository as a source, Command Central does not support selecting specific products from the image repository to include in the mirror. The mirror repository will always include all products from the image. You must omit the `artifacts` argument from the command when the source repository is an image.

## 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-10.5\_EUR\_Local" on the local installation for all products available in the "webMethods-10.5\_EUR" source repository that use the operating system of the local installation:

```
sagcc create repository products mirror name=webMethods-10.5_EUR_Local  
sourceRepos=webMethods-10.5_EUR
```

- To create a mirror repository with name "webMethods-10.5\_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-10.5_US_lnxamd64_w64  
nodeAlias=repoNode1 sourceRepos=webMethods-10.5_US_lnxamd64,  
webMethods-10.5_US_w64
```

- To create a mirror repository with name "webMethods-10.5\_US\_lnxamd64" in the local installation, with all products for the "LNXAMD64" platform from the image repository with name "webMethods-10.5\_US":

```
sagcc create repository products mirror name=webMethods-10.5_US_lnxamd64
sourceRepos=webMethods-10.5_US platforms=LNXAMD64
```

- To create a mirror repository with name "webMethods-10.5\_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-10.5_US_wMcore
nodeAlias=repoNode2 sourceRepos=webMethods-10.5_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.
[deleteImage={true   false}]	Optional. Whether to delete the image file referenced by the repository from the file system. Valid values: <ul style="list-style-type: none"> <li>■ true Delete the image file.</li> <li>■ false (default) Do not delete the image file.</li> </ul> When you do not include this argument, Command Central does not delete the image file from the file system.

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 <a href="#">“Common Options” on page 193</a> .

## 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
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### 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 202](#).

## 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
<i>host=install_server</i>	Required. The host name or IP address of the system hosting the repositories. If you do not specify a value, Command Central goes to the Empower website.
[name= <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.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### 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]
[platform=OS_Id] [options]
```

- Not supported by Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>repo_name</i>	Required. Specifies the name of the repository from which you want to generate a fix image archive.
dest= <i>filename.zip</i>	Required. A name for the generated fix image archive.
artifacts= <i>fixName1[_version]</i> , <i>fixName2[_version]</i>	Required. The names of the fixes to include in the image file. Specifying the fix version is optional. <div data-bbox="768 1472 1362 1696" style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <p><b>Important:</b> 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>
[platform= <i>OS_Id</i> ]	Optional. The ID of the operating system for which to export files.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For

Argument or Option	Description
	a description of the options, see <a href="#">“Common Options”</a> on page 193.

## 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 products export

Generates a product image archive from a product repository.

### Syntax

- Command Central syntax:

```
sagcc exec repository products export repo_Name
dest=filename.zip artifacts={productId[_version],
productId2[_version]|ALL} platform=OS_Id [options]
```

- Not supported by Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>repo_name</i>	Required. Specifies the name of a local product repository from which you want to generate a product image archive.
dest= <i>filename.zip</i>	Required. A name for the generated product image archive.
artifacts={ <i>productId[_version]</i> , <i>productId2[_version]</i>  ALL}	Required. The names of the products to include in the image file. Specifying the product version is optional. If you set the argument to ALL, Command Central exports all products from the product repository into the image archive.
platform= <i>OS_Id</i>	Required. The operating system ID of the products to include in the image archive. You can include products for one operating system in the product image archive.

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 <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- When exporting a list of specific product IDs, specified in the `artifacts` argument, JDK, license file, and the Installer libraries are not added to the exported image archive, though they are required to install products in a new installation directory. The exported image archive includes *only* the required product plug-ins for the specified products. It does not include all product plug-ins normally included in the Platform Manager`ALL` distribution. The exported image archive does not include Update Manager as a dependency.
- If you want to use a product mirror repository for the exported image archive, you must specify only mirror repositories created on the local Platform Manager installation.
- When listing a product in the `artifacts` argument, you must enclose the `productId` in single quotes (") if the ID includes underscore (\_), followed by a number. The single quotes ensure that Command Central does not interpret the (\_) character in the `productId` as the version delimiter in [`artifacts=productId[_version]`]. For example, to list the product with ID "WmPSFT\_E1\_898" and version 8.98.6282010.0.0, enter `artifacts='WmPSFT_E1_898'` (without the optional version) or `artifacts='WmPSFT_E1_898'_.8.98.6282010.0.0` (with the version).

## Example When Executing on Command Central

To generate a product image archive with name "myproducts.zip" that contains products with IDs "SPM" and "OSGI" for the operating system with ID "LNXAMD64" from a local product repository with name "productRepo":

```
sagcc exec repository products export productRepo dest=myproducts.zip
artifacts=SPM,OSGI platform=LNXAMD64
```

## sagcc exec repository mirror validateContent

Validates the integrity of a mirror repository against the source repository from which you created the mirror. When the contents of the mirror repository are not consistent, for example files are corrupted or missing, the command returns an ERROR status.

### Syntax

- Command Central syntax:

```
sagcc exec repository {products | fixes} mirror repoName validateContent
nodeAlias=nodeAlias [options]
```

- Not supported on Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>repoName</i>	Required. The name of the mirror repository to be validated.
<i>nodeAlias=</i> <i>nodeAlias</i>	Required. The alias name of the installation where the repository is stored.
[ <i>options</i> ]	The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Examples When Executing on Command Central

To validate the product mirror repository with name "webMethods-10.3-products" that is stored on the installation with alias "local":

```
sagcc exec repository products mirror webMethods-10.3-products validateContent nodeAlias=local
```

To validate the fix mirror repository with name "webMethods-10.3-fixes" that is stored on the installation with alias "local":

```
sagcc exec repository fixes mirror webMethods-10.3-fixes validateContent nodeAlias=local
```

## 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 <a href="#">“Common Options” on page 193</a> .

## 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
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

---

## 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 https://rubicon:8490/cce
```

```
sagcc list repository fixes --server https://rubicon:8490/cce
```

- To retrieve the available repository locations on a Command Central server, so that you can verify if you can access a repository over HTTP, HTTPS, or both:

```
sagcc list repository products properties=displayName,type,location,httpsLocation
```

```
sagcc list repository fixes properties=displayName,type,location,httpsLocation
```

For more information about connecting to repositories over HTTPS, see [“Using HTTPS to Connect to Repositories”](#) on page 373.

## 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
host= <i>install_server</i>	Required. The host name or IP address of the system hosting the repositories. If you do not specify a value, Command Central goes to the Empower website.
[name= <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.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options”</a> on page 193.

### 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=productId1_[version],productId2_[version|INSTALLED]]
[nodeAlias=nodeAlias] [options]
```

- Not supported by Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>repo_name</i>	Required. The name of the fix repository for which you want to list available fixes.
[products= <i>productId1_[version],productId2_[version INSTALLED]</i> ]	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 <a href="#">sagcc list inventory components</a> command.  When you specify <code>INSTALLED</code> Command Central lists the fixes that match the operating system and installed products on the target installation.
[nodeAlias= <i>nodeAlias</i> ]	Optional. Specifies the alias name of a Platform Manager installation. You can determine installation alias names using the <a href="#">sagcc list landscape nodes</a> command.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see “ <a href="#">Common Options</a> ” on page 193.

### 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] | FixId1,FixId2}
[products=productId1[_version],productId2[_version|INSTALLED] [nodeAlias=nodeAlias]
[showOlder={true|false}] [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]   FixId1,FixId2}	Required. The name or fix ID of the fix or fixes for which you want to check dependencies.  If you specify a fix name, you can optionally specify the fix version.  The fix ID is the official ID of the fix on Empower.

Argument or Option	Description
<code>[products=product1_[version],product2_[version]] INSTALLED]</code>	Optional. The IDs and version of the products for which to check fix dependencies. Use commas to separate each product ID. When you specify <code>INSTALLED</code> Command Central lists the fixes that match the operating system and installed products on the specified Platform Manager installation.
<code>[nodeAlias=nodeAlias]</code>	Optional. Specifies the alias name of a Platform Manager installation, for which to check the fix dependencies. When you omit this argument, the command checks the fix dependencies for the local installation.
<code>[showOlder={true false}]</code>	Optional. Specifies whether to list the old versions of the dependent fixes. Valid values: <ul style="list-style-type: none"> <li>■ <code>true</code> - list the old versions of the dependent fixes.</li> <li>■ <code>false</code> - (default) list only the latest version of the dependent fixes.</li> </ul>
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

When `showOlder=true`, you must also specify `products=INSTALLED`.

## 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
```

To list the old versions of the dependent fixes for the fix with ID “SPM\_10.5\_Core\_Fix4” for all products installed on the node with alias “node01” in the fix repository with name “repo1”:

```
sagcc list repository fixes dependencies repo1 SPM_10.5_Core_Fix4 products=INSTALLED
nodeAlias=node01 showOlder=true
```

## 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
repo_name	Required. The name of the fix repository that contains the fix.
fix_name[_version]	Required. The name of the fix for which you want to retrieve the readme. Specifying the fix version is optional.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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=productId_[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
repo_name	Required. The name of the product repository for which you want to list available products.
[products=productId_[version],productId2_[version INSTALLED]	Optional. The IDs and version of the products to list. Use commas to separate each product ID.

Argument or Option	Description
	<p>You can determine the IDs for run-time components using the <a href="#">sagcc list inventory components</a> 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 <a href="#">sagcc list landscape nodes</a> command.
<code>[type=product langpack ALL]</code>	Optional. Indicates whether to list only products, only language packages, or products and language packages.
<code>[languages=language1, language2]</code>	Optional. Indicates the languages for which to list language packages. Use comma as list separator.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

When listing a product in the `artifacts` argument, you must enclose the `productId` in single quotes (") if the ID includes underscore (\_), followed by a number. The single quotes ensure that Command Central does not interpret the (\_) character in the `productId` as the version delimiter in `[artifacts=productId[_version]]`. For example, to list the product with ID "WmPSFT\_E1\_898" and version 8.98.6282010.0.0, enter `artifacts='WmPSFT_E1_898'` (without the optional version) or `artifacts='WmPSFT_E1_898' _8.98.6282010.0.0` (with the version).

## 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.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### 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>[<del>products=productid1_[version],productid2_[version]</del>   <i>INSTALLED</i>]</code>	Optional. The IDs and version of the products for which to retrieve available languages. Use commas to separate each product ID.  You can determine the IDs for run-time components using the <a href="#">sagcc list inventory components</a> command.  When you specify <i>INSTALLED</i> Command Central lists the products that match the operating system and the products installed on the target installation.
<code>[nodeAlias=nodeAlias]</code>	Required. Specifies the alias name of a Platform Manager installation. You can determine installation alias names using the <a href="#">sagcc list landscape nodes</a> command.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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
<i>repo_name</i>	Required. The name of the repository that you want to update.
[credentials=credAlias]	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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with <a href="#">“sagcc create configuration data” on page 233</a> .
[location=image_file   repositoryURL]	Optional. The name of the image file or valid URL to the new location of the image file referenced by the repository.  For mirror repositories, the only valid value is [location=repositoryURL]
[sourceRepos=repo_name1,repo_name2...repo_nameN]	Optional. A list of the source repositories from which to update the mirror repository.
[productRepos=repo_name1,repo_name2...repo_nameN]	Optional. A list of product repositories for which to add fixes in the fix mirror repository.

Argument or Option	Description
[ <i>products=product_list</i>   <i>productId_version_list</i>   INSTALLED] ]	Optional. A list of IDs of the products for which to update the fixes in the mirror repository.
[ <i>artifacts=fix_list</i>   <i>fixId_version_list</i>   LATEST   ALL]	Optional. A list of the names of the fixes from the source repositories that you want to update in the mirror repository.
[ <i>description="description"</i> ]	Optional. The new description of the repository.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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
<i>repo_name</i>	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 <a href="#">sagcc list configuration instances</a> . If the configuration instance does not exist, you must create it with <a href="#">“sagcc create configuration data”</a> on page 233.
<code>[location=image_file   repositoryURL]</code>	Optional. The name of the image file or valid URL to the new location 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>[artifacts=productId1,productId2...productIdN]</code>	Optional. A list of IDs of the products from the source repositories that you want to update in the mirror repository.
<code>[platforms=OS_Id1,OS_Id2...OS_IdN]</code>	Optional. A list of the IDs of the operating systems of the products included in the mirror repository.

Argument or Option	Description
	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.
[description="description"]	Optional. The new description of the repository.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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=LNxAMD64,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
```

## Resources Commands

### 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 <a href="#">“Common Options” on page 193</a> .

#### 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 219](#). The command specifies “secret” for the user’s password.

## Security Credentials Commands

### 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=node_alias	Required. Specifies the alias name of the installation for which you want to associate the security credentials.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .

Argument or Option	Description
[runtimeComponentId= <i>componentid</i> ]	<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 <a href="#">sagcc list inventory components</a>.</p> <p><b>Important:</b> When configuring the credentials used for communication between Command Central and the Platform Manager installation, set this argument as follows: runtimeComponentId=SPM-CONNECTION.</p>
authenticationType={NONE BASIC TRUSTED}	<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"> <li>■ NONE</li> <li>■ BASIC - use basic authentication</li> <li>■ TRUSTED - use trusted authentication</li> </ul>
[username= <i>username</i> ]	<p>Optional. Specifies the user name to use when the authentication type is set to BASIC.</p>
[password= <i>password</i> ]	<p>Optional. Specifies the password to use when the authentication type is set to BASIC.</p>
{--input   -i} <i>file</i> {.xml .json}	<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 <a href="#">“input” on page 204</a>.</p> <p><b>Tip:</b> To determine how to specify the data in the input file, use <a href="#">sagcc get security credentials</a> to retrieve data for existing security credentials. For example, if you want to use an XML input file, use <a href="#">sagcc get security credentials</a> with the <code>--format xml</code> option to retrieve the data in XML format.</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 <a href="#">“Common Options” on page 193</a>.</p>

## 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 215](#) and [“username” on page 219](#). 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.  If you omit this parameter, the command sets a global shared secret for the whole landscape.
secret= <i>text_string</i>	Specify a shared secret password that Platform Manager uses to encrypt or decrypt configuration passwords.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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=node_alias]	Optional. Specifies the alias name of the installation for which you want to delete security credentials.  You can view a list of installations and their aliases using <a href="#">sagcc list landscape nodes</a> .
[runtimeComponentId=componentid]	Optional. Specifies the run-time component for which you want to delete security credentials.  You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
[options]	Optional. Refer to the command syntax for a list of the options the command supports. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 exec security encrypt

The command encrypts user passwords for product configurations, using the environment or global shared secret configured in Command Central. You can copy the encrypted value from the command output and use it in place of the clear-text password, for example when you must specify this password in a template or properties file.

### Syntax

- Command Central syntax:

```
sagcc exec security encrypt password=password
[sharedSecret=sharedSecret] [options]
```

- Not supported on Platform Manager

### Arguments and Options

Argument or Option	Description
<code>password=<i>password</i></code>	Required. The user password to encrypt. Valid value is any string.
<code>[<i>sharedSecret=sharedSecret</i>]</code>	Optional. The shared secret configured for the Command Central environment. If you omit this parameter, the command uses the global shared secret, configured for Command Central. For more information about configuring a shared secret, see <a href="#">“Configure Shared Secrets to Encrypt Instance and Component Passwords”</a> on page 45 and <a href="#">“sagcc add security credentials sharedsecret”</a> on page 417.
<code>[<i>options</i>]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options”</a> on page 193.

## Usage Notes

When you create a new environment with templates, use the same shared secret in the new environment as the shared secret you specified when encrypting the password. For example, if you encrypted the password with the global shared secret, use the global shared secret for the target installation in the new environment. If you encrypted the password with a custom shared secret for the environment, use the same custom shared secret on the target installation.

## Examples When Executing on Command Central

To encrypt the proxy configuration password "manage", using the global shared secret configured for Command Central:

```
sagcc exec security encrypt password=manage
```

To encrypt the db password "manage\_123", using the "secret" shared secret, configured for the Command Central environment:

```
sagcc exec security encrypt password=manage_123 sharedsecret=secret
```

## sagcc exec security encrypt input

The command encrypts user passwords for product configurations included in a properties file, using the environment or global shared secret configured in Command Central. The command encrypts the value of a property in the input file only when the property name is prefixed with `@secure`, for example `@secure.password=value`. You can use the encrypted values from the command output in place of clear text passwords when performing bulk operations with multiple passwords. For example, you can copy the command output into a template properties file.

## Syntax

- Command Central syntax:

```
sagcc exec security encrypt [--input|-i] fileName.properties  
[sharedSecret=sharedSecret] [options]
```

- Not supported on Platform Manager

## Arguments and Options

Argument or Option	Description
<code>{--input -i} fileName.properties</code>	Required. Specifies the properties file with the product configurations that have user passwords to encrypt. For information about the format of the properties file, see "Format of the Properties File". For information about the <code>{--input -i}</code> option, see <a href="#">"input" on page 204</a> .

Argument or Option	Description
[sharedSecret= <i>sharedSecret</i> ]	Optional. The shared secret configured for the Command Central environment. If you omit this parameter, the command uses the global shared secret, configured for Command Central. For more information about configuring a shared secret, see <a href="#">“Configure Shared Secrets to Encrypt Instance and Component Passwords”</a> on page 45 and <a href="#">“sagcc add security credentials sharedsecret”</a> on page 417.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options”</a> on page 193.

## Usage Notes

- Software AG recommends that you set the encoding of the properties file to UTF-8.
- By default, the command writes the output with the encrypted password values to the console. To write the command output to a file, include the [“output”](#) on page 209 option.
- When you create a new environment with templates, use the same shared secret in the new environment as the shared secret you specified when encrypting the passwords. For example, if you encrypted the passwords with the global shared secret, use the global shared secret for the target installation in the new environment. If you encrypted the passwords with a custom shared secret for the environment, use the same custom shared secret on the target installation.

## Format of the Properties File

When you want to encrypt a property in the properties file, you must prefix the property name with `@secure`` in the following format: `@secure.propertyName=value`. Note that the property cannot have a multi-line value. In the following example, if you prefix the password properties with `@secure`:

```
# the user name of the administrator user
username=Administrator
# the password of the administrator user
@secure.adminpassword=cleartext_password
# the user name of the database user
username=dbuser
# the password of the database user
@secure.dbpassword=cleartext_password
```

The command output will return the following:

```
# the user name of the administrator user
username=Administrator
```

```
# the password of the administrator user
adminpassword=encrypted_value
# the user name of the database user
username=dbuser
# the password of the database user
dbpassword=encrypted_value
```

## Examples When Executing on Command Central

To encrypt all password values with the prefix @secure contained in the "prodconfig.properties" file, using the global shared secret configured for Command Central:

```
sagcc exec security encrypt -i prodconfig.properties
```

To encrypt all password values with prefix @secure contained in the "prodconfig.properties" file, using the global shared secret configured for Command Central and write the output to a file named "encryptedPasswords.properties":

```
sagcc exec security encrypt -i prodconfig.properties -o encryptedPasswords.properties
```

## 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= <i>node_alias</i> ]	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 <a href="#">sagcc list landscape nodes</a> .
[runtimeComponentId= <i>componentid</i> ]	Optional. Specifies the run-time component for which you want to retrieve security credentials.  You can determine the IDs for run-time components using <a href="#">sagcc list inventory components</a> .
[ <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 “ <a href="#">Common Options</a> ” on page 193.

## 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
```

## Stacks and Layers Commands

### sagcc create stacks

Creates a new product stack.

#### Syntax

- Command Central syntax:

```
sagcc create stacks alias=stackName release=version
[description=description] [platform=platformCode] [options]
```

- Not applicable to Platform Manager.

#### Arguments and Options

Argument or Option	Description
<i>alias=stackName</i>	Required. The name of the stack. Specify a name that is unique among all stacks this Command Central manages.
<i>release=version</i>	Required. The release version of the stack. For example, 10.1
<i>description=description</i>	Optional. A description of the new stack. If you type a description with spaces, place quotes around the description.
[ <i>platform=platformCode</i> ]	Optional. The ID of the operating system of the stack.  When you do not include this parameter, the command uses the operating system on the localhost machine.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

#### Usage Notes

- Command Central automatically assigns an index number to a new stack. The index determines the order of positioning of stacks on the Stacks screen in the Command Central web user interface. You can change the index number of a stack using the [“sagcc update stacks index” on page 447](#) command.
- 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", for installations running on Linux:

```
sagcc create stacks alias=stack01 release=10.1 description="test environment stack"
platform=lnxamd64 --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 auto

Generates product stacks with layers automatically from existing environments managed by one Command Central. You can also use the command to generate product stacks only for a specific environment, or only for product installations that have the same release version. Command Central compares the existing installations and:

- Includes in a generated stack only installations that have equal:
  - Operating systems
  - Release versions
  - Installation directory location and name
  - Platform Manager port number and protocol (HTTP or HTTPS). If an installation does not match all of the criteria, Command Central includes that installation in a separate stack.
- Includes in a generated run-time layer only the products from the existing installations that have equal run-time component IDs. If a run-time component ID does not match the IDs of the other run-time components in the layer, Command Central includes the run-time component in a separate layer.

## Syntax

- Command Central syntax:

```
sagcc create stacks auto [environment=env_alias] [release=version]
[stackPrefix=stackPrefix] [options]
```

- Not applicable to Platform Manager.

## Arguments and Options

Argument or Option	Description
<code>[environment=env_alias]</code>	Optional. The alias of the environment, for which to generate stacks and layers.

Argument or Option	Description
	If you do not specify an environment or do not include this argument, Command Central generates stacks and layers for all environments.
[ <i>release=version</i> ]	Optional. The release version of the nodes for which to generate stacks and layers. For example, 10.7  Use this argument to filter by version the installations that you want to include in the generated stacks. If you do not specify a release version or do not include this argument, Command Central generate stacks and layers for all release versions in the existing installations.
[ <i>stackPrefix=stackPrefix</i> ]	Optional. Specifies a prefix to add to the names of the generated stacks. By default, the names of the generated stacks are "Stack1", "Stack2", ... "StackN". Use this argument to add a unique prefix to the names of all generated stacks.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Limitations

Command Central does not automatically:

- Generate the database layers. You must add the required database layers to the generated stacks.
- Include the local installation in the generated stacks. Command Central is typically installed in a different installation directory from the installations that it manages and is not included in the product stacks.
- Add the same installation in more than one stack. If an existing installation is already in a product stack, whether generated automatically or created from scratch, you cannot add the installation to another stack.

## Examples

To create stacks and layers for all environments and installations that the local Command Central manages:

```
sagcc create stacks auto
```

To create a stack for the installations with version "10.5" in all environments that the local Command Central manages:

```
sagcc create stacks auto release=10.5
```

To create stacks for the installations with version "10.5" in the "Dev" environment and add the "stable" prefix to the names of all generated stacks:

```
sagcc create stacks auto environment=Dev release=10.5 stackPrefix=stable
```

## 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 INFRA-LOCAL is the alias of the instance with ID CCE-LAYER-TYPES-infra-local. 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=<i>repoName</i></code> parameter to point to the bootstrap installer from which to provision Platform Manager
[ <i>templateProperty1=templateValue1...</i> ]	Optional. User-defined parameters for the micro template referenced in the layer definition.

Argument or Option	Description
<code>description=description</code>	Optional. A description of the new stack. If you type a description with spaces, place quotes around the description.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

Command Central automatically assigns an index number to a new layer. The index determines the order of positioning of layers in a stack in the Command Central web user interface. You can change the index number of a layer using the [“sagcc update stacks layers index” on page 448](#) command.

## 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
<code>stackName</code>	Required. The name of the stack to which the layer belongs.

Argument or Option	Description
<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 <a href="#">“Common Options” on page 193</a> .

## 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. If the stack contains layers, you can also delete all layers in the stack.

### Syntax

- Command Central syntax:

```
sagcc delete stacks stackName [deleteLayers={true|false}] [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.
[ <i>deleteLayers</i> ={true false}]	Optional. Specifies whether the layers in the stack will be deleted. Valid values: <ul style="list-style-type: none"> <li>■ true - Delete the layers.</li> <li>■ false (default) - Do not delete the layers.</li> </ul>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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.
- If you set `deleteLayers=false` and attempt to delete a stack that contains layers, the command will not delete the stack and will return an error message.

## Example

To delete the stack with name "stack01" and all layers in the stack:

```
sagcc delete stacks stack01 deleteLayers=true
```

## 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 <a href="#">“Common Options” on page 193</a> .

## 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.
- You cannot delete the infrastructure layer before you delete all other layers in the stack. If you attempt to delete the infrastructure layer, but the stack contains other layers, the command will return an error message.

## Example

To delete a layer with name "layer01" in the stack with alias "stack01":

```
sagcc delete stacks stack01 layers layer01
```

## sagcc exec stacks migrate

Upgrades product installations with version 10.3 and higher using a stack. You can use an upgrade stack for one of the following supported migration types:

- Overinstall: same installation directory, hosts, and ports
- Side-by-side: new installation directory, same hosts and ports
- Cross-host: new hosts, same ports. The installation directory can have the same or a new value.

Note that the migration template that the upgrade stack uses determines which migration type you can run. For details about generating migration templates, see [“sagcc exec templates composite generate migration” on page 472](#).

## Syntax

- Command Central syntax:

```
sagcc exec stacks stackName migrate migration.type={overinstall|sidebyside|crosshost}
[-i input-parameters-file.properties] [param1=value1 param2=value2 ...]
[migration.reuse.templates={true|false}] [migration.dry.run={true|false}] [options]
```

- Not applicable to Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack to use to migrate the product installations.
migration.type={overinstall sidebyside crosshost}	Required. The type of migration to run when upgrading the product installations. Valid values: <ul style="list-style-type: none"> <li>■ overinstall</li> <li>■ sidebyside</li> <li>■ crosshost</li> </ul>
[-i <i>input-parameters-file.properties</i> ]	Optional. The properties file that contains the required and optional properties, for which you should specify values during migration.

Argument or Option	Description
[param1=value1 param2=value2 ...]	Optional. Include the required and optional properties, for which you should specify values during migration, as arguments in the command.
[migration.reuse.templates={true false}]	<p>Optional. After a failed migration attempt, you can include this argument in the command when you next attempt to migrate with the same stack. Indicates whether to use the generated templates from the last failed migration run in the next run. Valid values:</p> <ul style="list-style-type: none"> <li>■ true - reuses the generated templates</li> <li>■ false (default) - does not reuse the generated templates</li> </ul>
[migration.dry.run={true false}]	<p>Optional. Indicates whether Command Central should try a dry run of the upgrade. In the dry run, Command Central generates templates, validates the templates and the migration properties, but does not apply the templates on the target nodes. You can review the results of the dry-run job and check the templates before the actual migration run. Valid values:</p> <ul style="list-style-type: none"> <li>■ true - generate and validate the templates, but do not apply them</li> <li>■ false - (default) - generate, validate, and apply the templates</li> </ul> <p>When you do not include the argument, Command Central does not try a dry run. For information about how Command Central validates a template, see <a href="#">“Validating a Composite Template” on page 129</a>.</p>
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- To obtain the required and optional properties for migration, run the `sagcc list stacks migrate properties` command. You can also use the **Stacks > Upgrade Stack** web user interface to export the properties for an existing stack into a properties file to use as input file for the `stacks migrate` command.
- You must specify the migration properties either as arguments in the command, or in an input file. You can also include both `[param1=value1 param2=value2 ...]` and `[-i <input-parameters-file>.properties]` in the same command. The values specified in the arguments will override the values of the parameters specified in the properties file.

- If a migration attempt fails, run the `sagcc list stacks migrate status` command to check the migration status of the stack. The command output also indicates whether any templates were generated during the failed attempt.
- When running the `stacks migrate` command, the command processes the `fixes` property as follows:
  - `fixes=ALL` By default, installs all the latest fixes for the respective layer, which are available in the specified fix repository for the products in the target installations.
  - `fixes=` If you leave the value empty, the command does not install fixes on the target installation.
  - `fixes=[fixName1[_version], fixId2, ..., fixIdN]` If you include this argument, the command installs only the fixes listed in the argument. To retrieve a list of fix names or fix IDs, run [“sagcc list repository fixes content” on page 402](#).

Software AG recommends that you install at least fix version 1 for each product, because the `_Fix1` fixes typically address important technical changes.

## Example

To run an overinstall upgrade for a product installation using the product stack named "stack01" and the properties specified in the "stack01-migrate.properties" file:

```
sagcc exec stacks stack01 migrate migration.type=overinstall
-i stack01-migrate.properties
```

## Related Topics

[“sagcc list stacks migrate properties” on page 442](#)

[“sagcc list stacks migrate status” on page 445](#)

## sagcc list stacks

Lists all stacks that Command Central manages. Information about the stacks can include:

- Index number
- 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
<i>[options]</i>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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:

- Index number
- 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.

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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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 get stacks layers fixes compare

Compare information about fixes that have been applied to products in a layer.

### Syntax

- Command Central syntax:

```
sagcc get 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 <a href="#">"Common Options" on page 193</a> .

### 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 <a href="#">“Common Options” on page 193</a> .

## 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.
<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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

## 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 <a href="#">“Common Options” on page 193</a> .

### 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 migrate properties

Lists the required and optional migration properties for a stack. Required are properties, for which you must specify values when migrating product installations. For example, product and fix repositories are required properties. Optional are properties, for which you can keep the current value or specify a new value. Information about the properties can include:

- Layer - The name of the layer, which uses the property.
- Name - Describes the function of the property.
- Key - The parameter name of the property. For properties that have the same parameter name in different layers of the stack, the key starts with the name of the layer. For example "infrastructure.fixes" is the key for the fixes parameter in the infrastructure layer; "database.fixes" is the key for the fixes parameter in the database layer.
- Value - The value of the property. The values of the required properties are empty, \${}. The optional properties have a specific value.

The migration properties that the command returns will depend on the type of migration you want to run. You can list migration properties for the following migration types:

- Overinstall: same installation directory, hosts, and ports
- Side-by-side: new installation directory, same hosts and ports
- Cross-host: new hosts, same ports. The installation directory can have the same or a new value.

## Syntax

- Command Central syntax:

```
sagcc list stacks stackName migrate properties
migration.type={overinstall|sidebyside|crosshost}
[-f application/properties] [options]
```

- Not applicable to Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack to use to migrate the product installations.
migration.type={overinstall sidebyside crosshost}	Required. Specifies the type of migration to execute when upgrading product installations. Valid values: <ul style="list-style-type: none"> <li>■ overinstall</li> <li>■ sidebyside</li> <li>■ crosshost</li> </ul>
[-f application/properties]	Optional. Returns a list of the required and optional properties in the <parameter>=<value> format. The required properties have an empty value, for example <code>cc.installer=\${}</code> . The optional properties are listed with their current value, for example <code>nodes.ssh.port=22</code> .
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- When you run the command without the `-f application/properties` option, the command output is in tabular format. When you include the `-f application/properties` option, the output is in the <parameter>=<value> format, which is easier to transfer to a properties file, or as an argument on the command line.
- You can add the properties from the command output in a properties file, which you can use with the `sagcc exec stacks migrate` command or to import the properties into a stack, using the **Stacks** > Upgrade Stack web user interface.
- The command returns the `fixes` property for all layers in the stack with the value `ALL`. When `fixes=ALL`, Command Central installs all the latest fixes for the respective layer, which are available in the specified fix repository for the products in the target installations. For more

information about using the `fixes` property when migrating the stack, see the "Usage Notes" in ["sagcc exec stacks migrate"](#) on page 432.

## Examples

To list the migration properties for the stack with name "stack01", when running an overinstall upgrade:

```
sagcc list stacks stack01 migrate properties migration.type=overinstall
```

Layer	Name	Key
Value		
Infrastructure	Command Central Bootstrap Installer	cc.installer
	{}	
Infrastructure	Product Repository	repo.product
	{}	
Infrastructure	Fix Repository	repo.fix
	{}	
Infrastructure	Platform Manager Credentials Key	nodes.spm.credentials.key
	{}	
Infrastructure	Remote Machine SSH Port	nodes.ssh.port
	22	
Infrastructure	Infrastructure Fixes	infrastructure.fixes
	ALL	
	...	

To list the migration properties for the stack with name "stack01" as key-value pairs, when running a side-by-side upgrade:

```
sagcc list stacks stack01 migrate properties migration.type=sidebyside
-f application/properties
```

```
nodes.spm.credentials.key={}
repo.fix={}
nodes.ssh.substitute.credentials.key=${nodes.ssh.login.credentials.key}
cc.installer={}
nodes.ssh.port=22
infrastructure.fixes=ALL
nodes.ssh.login.credentials.key={}
repo.product={}
```

To add the migration properties for the stack with name "stack01" as key-value pairs in a properties file with name "stack01-migration.properties", when running a cross-host upgrade:

```
sagcc list stacks stack01 migrate properties migration.type=crosshost
-f application/properties -o stack01-migration.properties
```

## Related Topics

["sagcc exec stacks migrate"](#) on page 432

["sagcc list stacks migrate status"](#) on page 445

## sagcc list stacks migrate status

Use this command to check the migration status after a failed migration attempt with a stack. The command output also indicates whether Command Central generated any templates before the migration attempt failed.

### Syntax

- Command Central syntax:

```
sagcc list stacks stackName migrate status [options]
```

- Not applicable to Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which to return migration status.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Output

Column	Description
Stack Alias	The alias name of the stack.
Templates Generated	Conditional. Indicates whether Command Central generated any templates during the failed migration run. Valid values: <ul style="list-style-type: none"> <li>- true - Command Central has generated one or more templates</li> <li>- false - no templates generated during the failed run</li> </ul>
Resume Previous Run	Conditional. Indicates whether Command Central will resume the migration from the point of failure or will start over in the next migration run with the same stack. <ul style="list-style-type: none"> <li>- true - will resume the migration from the point of failure</li> <li>- false - will start the migration from the beginning</li> </ul>

## Usage Notes

- If both "Templates Generated" and "Resume Previous Run" are true, Command Central will reuse the generated templates when re-running the migration.
- If "Templates Generated" is true and "Resume Previous Run" is false, by default Command Central will generate new templates and not reuse the generated templates from the failed run. If you want to use the templates generated in the failed run, override the default behavior by including `migration.reuse.templates=true` when running the `sagcc exec stacks migrate` command to migrate the same stack again. For example, you can include this argument when you have edited and re-imported one or more of the generated templates from the failed run. You can find the generated templates in the Templates view of the Command Central web user interface.

## Example

To check the migration status of the product stack named "stack01":

```
sagcc list stacks stack01 migrate status
```

Stack Alias	Templates Generated	Resume Previous Run
stack 01	true	false

## Related Topics

“sagcc exec stacks migrate” on page 432

“sagcc list stacks migrate properties” on page 442

## sagcc get stacks layers products compare

Compares the products that belong to a layer.

## Syntax

- Command Central syntax:

```
sagcc get 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 <a href="#">“Common Options” on page 193</a> .

## 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.

## sagcc update stacks index

Updates the index number of the stack. The index determines the order of positioning of stacks on the Stacks screen in the Command Central web user interface.

## Syntax

- Command Central syntax:

```
sagcc update stacks stackName index index [options]
```

- Not applicable to Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The name of the stack for which to update the index number.
<i>index</i>	Required. The new index number of the stack. Valid values are positive integers equal to or larger than 0.

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 <a href="#">“Common Options” on page 193</a> .

## Example

To update the index number of the stack with name "stack01" to 2:

```
sagcc update stacks stack01 index 2
```

## sagcc update stacks layers index

Updates the index number of the layer. The index determines the order of positioning of layers in a stack in the Command Central web user interface.

### Syntax

- Command Central syntax:

```
sagcc update stacks stackName layers layerName
index index [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 update the index number.
<i>index</i>	Required. The new index number of the layer. Valid values are positive integers equal to or larger than 1.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Usage Notes

The index number of the infrastructure layer is always 0 and you cannot update it. No other layer can have an index number of 0.

## Example

To update the index number of the layer with name "layer02" from the stack with name "stack01" to 3:

```
sagcc update stacks stack01 layer02 index 3
```

## sagcc update stacks alias

Updates the name of the specified stack.

### Syntax

- Command Central syntax:

```
sagcc update stacks stackName alias newStackName [options]
```

- Not applicable to Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>stackName</i>	Required. The current name of the stack whose name to update.
<i>newStackName</i>	Required. The new name to assign to the stack.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Examples

To update the name of the stack "Stack1" to "Test1":

```
sagcc update stacks Stack1 alias Test1
```

## sagcc update stacks layers alias

Updates the name of the specified layer.

### Syntax

- Command Central syntax:

```
sagcc sagcc update stacks stackName layers layerName alias  
newLayerName [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 current name of the layer whose name to update.
<i>newLayerName</i>	Required. The new name to assign to the layer.
[options]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Examples

In the stack with name "Test1", to update the name of the "OSGI-CCE" layer to "CommandCentral-Test1":

```
sagcc update stacks Test1 OSGI-CCE alias CommandCentral-Test1
```

## Template Commands

---

### 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 <a href="#">“Common Options” on page 193</a> .

## 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. After you run the `apply composite templates` command, Command Central validates the composite template as a first processing step. See the "Skipping Validation Checks" section in this topic for the arguments you can use to skip some of the template validation.

### Syntax

- Command Central syntax:

- With input parameters:

```
sagcc exec templates composite apply template_alias
environment.mode={provision|migration|migrate|maintenance}
[environment.type=type ]
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={provision migrate migration maintenance}</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"> <li>■ <code>provision</code> - installs a new environment</li> <li>■ <code>migrate migration</code> - migrates an existing environment</li> <li>■ <code>maintenance</code> - installs only the fixes defined in the template. For example, you can use this mode to install a Database Component Configurator fix that contains database schema changes.</li> </ul>
<code>[environment.type=type]</code>	Optional. Specify what type of environment you want to create. You must specify one of the environment types defined in the <code>template_alias</code> YAML file.
<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 <a href="#">“Common Options” on page 193</a> .

## 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 dependencies are available in the repositories.

Argument or Option	Description
<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 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.
- For details about how Command Central applies a database micro template or a composite template with a database layer, based on the value of the `environment.mode` argument, see [“Managing Database Components Using a Template” on page 152](#).
- 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*.
- If a sub-job of the main composite template apply job fails, the failure of the sub-job might cause the main template apply job to fail with an `ERROR`.
- When executing the command by specifying an expected status value with the `{--expected-value | -e}` option, the CLI output contains a non-zero return code if the expected status value is not retrieved and a zero return code if the expected status is successfully retrieved.
- When executing the command by specifying an encrypted value in a password argument, enclose the value in `\` to ensure that the encrypted value is in quotes when it is processed as the value of a password parameter in the template YAML file.

## 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
<code>{--input   -i}</code> <code>filename.{zip yaml}</code>	Required. An input file that contains a composite template. For more information about this command option, see <a href="#">“input” on page 204</a> .
<code>overwrite={true   false}</code>	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"> <li>■ <code>true</code> - delete the template</li> <li>■ <code>false</code> - (default) do not delete the template</li> </ul>
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 import repository

Imports all or selected template assets from the default template asset repository into the local Command Central installation.

For example, after you connect to the Default Templates on GitHub repository, you can use the command to update/import the latest changes from the template asset repository on GitHub into the default template assets repository on Command Central.

## Syntax

- Command Central syntax:

```
sagcc exec templates composite import assetsRepoName
[templateNames=TemplateName1,TemplateName2,...]
[applyLayerDefinitions={true|false}] [overwrite={true|false}] [options]
```

- Not supported on Platform Manager

## Arguments and Options

Argument or Option	Description
<i>assetsRepoName</i>	The name with which you registered the default template asset repository in Command Central.
[ <i>templateNames=TemplateName1,TemplateName2,...</i> ]	Optional. A list of the aliases of the templates to import. If you do not specify this argument, all templates in the template assets repository on GitHub will be imported.
[ <i>applyLayerDefinitions={true false}</i> ]	Optional. Specifies whether to create layer type definitions for the imported templates. Command Central creates layer type definitions only for the default templates used in one of the CCE-LAYER-TYPES-* definitions in the sag-cc-layer-defs template. Valid values: <ul style="list-style-type: none"> <li>■ <i>true</i> - (default) creates layer type definitions for the imported templates</li> <li>■ <i>false</i> - does not create layer type definitions</li> </ul>
[ <i>overwrite={true false}</i> ]	Required only when updating a template assets repository of type "Default Templates on GitHub". If a template is updated on GitHub, it will be re-imported in the template assets repository (registered in Command Central) with the same template alias. The <i>overwrite</i> argument controls whether the command will overwrite the template of the same alias in the registered repository. Valid values: <ul style="list-style-type: none"> <li>■ <i>true</i> - overwrite the templates in the registered repository</li> <li>■ <i>false</i> - (default) do not overwrite the templates</li> </ul>
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Examples When Executing on Command Central

To update the templates asset repository with name "DefaultTemplatesRepo" with the latest changes in the default templates in the template assets repository on GitHub and overwrite the templates of the same alias in the "DefaultTemplatesRepo":

```
sagcc exec templates composite import DefaultTemplatesRepo overwrite=true
```

## sagcc exec templates composite generate

Based on the input arguments that you include, the command can:

- Generate a run-time micro template from an installed product instance.
- Generate an infrastructure micro template from the Platform Manager instance in an installation.
- Export the database for the products in an installation to a database micro template.
- Generate a migration template from an installed product instance.

Command Central imports the generated template automatically and creates a layer type definition for the template in the local Command Central installation. You can use this layer type definition to create a layer for the product instance in an existing product stack. See the [“sagcc create stacks layers” on page 428](#) command.

After you run the command, you can monitor the progress of the template generation job from the Jobs view in the Command Central web user interface.

## Syntax

- Command Central syntax:

- Export an installed product instance to a run-time micro template

```
sagcc exec templates composite generate alias=templateAlias nodeAlias=nodeAlias
runtimeComponentId=runtimeComponentId [parameterize={true | false}]
[paramPrefix=paramPrefix] [options=CONFIGURATION] [options=INSTANCES]
[options=PRODUCTS]
[options=FIXES] [options=INFRASTRUCTURE] [overwrite={true | false}]
[createLayerType={true | false}]
```

- Export the Platform Manager instance in an installation to an infrastructure micro template

```
sagcc exec templates composite generate alias=templateAlias nodeAlias=nodeAlias
options=INFRASTRUCTURE [parameterize={true | false}] [paramPrefix=paramPrefix]
[overwrite={true | false}] [createLayerType={true | false}]
```

- Export the database for the products in an installation to a database micro template

```
sagcc exec templates composite generate alias=templateAlias nodeAlias=nodeAlias
options=DATABASE db.url=JDBC_URL db.username=username
```

```
db.password=password db.type=type [overwrite={true|false}]
[dbUseComponentsLatestVersion={true|false}] [createLayerType={true|false}]
-s local_CCE_URL
```

- Not supported on Platform Manager.

## Arguments and Options for Exporting an Instance to a Run-time Template

Argument or Option	Description
<code>alias=templateAlias</code>	Required. The alias name of the run-time micro template to generate.
<code>nodeAlias=nodeAlias</code>	Required. The alias name of the installation on which the product instance is installed.
<code>runtimeComponentId=runtimeComponentId</code>	Required. The ID of the product instance for which to generate the template.
<code>[parameterize={true false}]</code>	Optional. Specifies whether to parameterize the exported configurations. Valid values: <ul style="list-style-type: none"> <li>■ <code>true</code> - parameterize the configurations</li> <li>■ <code>false</code> - (default) do not parameterize the configurations</li> </ul>
<code>[paramPrefix]</code>	Optional. Use only when <code>parameterize=true</code> . Specifies a prefix to be added to the exported parameter names.
<code>[options=CONFIGURATION]</code> <code>[options=INSTANCES]</code> <code>[options=PRODUCTS]</code> <code>[options=FIXES]</code>	Optional. Include one or more of the options arguments to specify what information about the product instance you want to include in the <code>templates: section</code> of the generated template: <ul style="list-style-type: none"> <li>■ <code>CONFIGURATION</code> - Includes the configuration for the product instance (and its child run-time components)</li> <li>■ <code>INSTANCES</code> - Includes the instance properties</li> <li>■ <code>PRODUCTS</code> - Includes a list of product inline templates in the <code>templates:templateAlias:products:</code> section. The command determines which products get included in the list based on the version of the source installation.</li> <li>■ <code>FIXES</code> - Includes a <code>templates:templateAlias:fixes:</code> section. Only with DSL "1.1", if the source product instance has been updated with patches, this option also includes a <code>templates:templateAlias:patches:</code> section. Based on the version of the source installation, the generated template will differ as described in the <code>options=FIXES</code> usage note.</li> </ul>

Argument or Option	Description
	For details about the options arguments, see the Usage Notes.
[options=INFRASTRUCTURE]	Optional. For details about using the argument when exporting a product instance, see the "Usage Notes for Exporting a Platform Manager Instance".
[overwrite={true false}]	Required when you execute the command to update a template. Specifies whether to overwrite an existing template with the same alias name. Valid values: <ul style="list-style-type: none"> <li>■ true - overwrite the template</li> <li>■ false (default) - do not overwrite the template</li> </ul>
[createLayerType={true false}]	Optional. Specifies whether to create a new RUNTIME layer type definition that references the generated template. The alias of the new layer type definition uses the alias of the generated template, for example if the template alias is my-template, the layer definition alias is MY-TEMPLATE. <ul style="list-style-type: none"> <li>■ true (default) - create a new layer definition</li> <li>■ false - do not create a new layer definition</li> </ul>

## Arguments and Options for Exporting Platform Manager to an Infrastructure Template

Argument or Option	Description
alias= <i>templateAlias</i>	Required. The alias name of the infrastructure micro template to generate.
nodeAlias= <i>nodeAlias</i>	Required. The alias name of the installation from which you export Platform Manager.
options=INFRASTRUCTURE	Required. Exports all data for the Platform Manager instance, such as Platform Manager components, product plugins, fixes, and configurations. You can also filter which Platform Manager data to include in the generated infrastructure template as described in the "Usage Notes for Exporting a Platform Manager Instance".
[parameterize={true false}]	Optional. Specifies whether to parameterize the exported configurations. Valid values: <ul style="list-style-type: none"> <li>■ true - parameterize the configurations</li> <li>■ false - (default) do not parameterize the configurations</li> </ul>

Argument or Option	Description
	Note that the <code>cc.installer</code> and <code>credentials</code> keys properties are parameterized in the generated template, even when <code>parameterize=false</code> , because they are required parameters.
<code>[paramPrefix]</code>	Optional. Use only when <code>parameterize=true</code> . Specifies a prefix to be added to the exported parameter names.
<code>[overwrite={true false}]</code>	Required when you execute the command to update a template. Specifies whether to overwrite an existing template with the same alias name. Valid values: <ul style="list-style-type: none"> <li>■ <code>true</code> - overwrite the template</li> <li>■ <code>false</code> (default) - do not overwrite the template</li> </ul>
<code>[createLayerType={true false}]</code>	Optional. Specifies whether to create a new INFRASTRUCTURE layer type definition that references the generated template. The alias of the new layer type definition uses the alias of the generated template, for example if the template alias is <code>my-template</code> , the layer definition alias is <code>MY-TEMPLATE</code> . <ul style="list-style-type: none"> <li>■ <code>true</code> (default) - create a new layer definition</li> <li>■ <code>false</code> - do not create a new layer definition</li> </ul>

## Arguments and Options for Exporting a Database to a Database Template

Argument or Option	Description
<code>alias=templateAlias</code>	Required. The alias name of the database micro template to generate.
<code>nodeAlias=nodeAlias</code>	Required. The alias name of the installation, from which to export the database.
<code>options=DATABASE</code>	Required. Exports the database for the products in the installation to a database template. For more details about how to use the generated database template, see <a href="#">“Managing Database Components Using a Template” on page 152</a> .
<code>db.url=JDBC_URL</code>	Required. The JDBC URL to use to connect to the database.
<code>db.username=username</code>	Required. The name of a user account that has access to the database.
<code>db.password=password</code>	Required. The password of the user account you specified in <code>db.username</code> .

Argument or Option	Description
<code>db.type=type</code>	Required. The type of the database to which you want to connect, such as <code>oracle</code> , <code>sqlserver</code> , or <code>mysql</code> . For a list of supported database types, see <i>System Requirements for Software AG Products</i> .
<code>[overwrite={true false}]</code>	Required when you execute the command to update a template. Specifies whether to overwrite an existing template with the same alias name. Valid values: <ul style="list-style-type: none"> <li>■ <code>true</code> - overwrite the template</li> <li>■ <code>false</code> (default) - do not overwrite the template</li> </ul>
<code>[dbUseComponentsLatestVersion={true false}]</code>	Optional. Indicates whether to export the database components and schemas with the latest version. Valid values: <ul style="list-style-type: none"> <li>■ <code>true</code> (default) - export the database components and schemas with the latest version</li> <li>■ <code>false</code> - export the database components and schemas with the release version of the source installation</li> </ul>
<code>[createLayerType={true false}]</code>	Optional. Specifies whether to create a new DATABASE layer type definition that references the generated template. The alias of the new layer type definition uses the alias of the generated template, for example if the template alias is <code>my-template</code> , the layer definition alias is <code>MY-TEMPLATE</code> . <ul style="list-style-type: none"> <li>■ <code>true</code> (default) - create a new layer definition</li> <li>■ <code>false</code> - do not create a new layer definition</li> </ul>
<code>-s</code>	Specify the URL of the local Command Central. For details about this option see <a href="#">“server” on page 215</a> .

## General Usage Notes

- All exported passwords are encrypted. For details about encrypting passwords, see [“Encrypting Passwords in Templates” on page 150](#).
- The command allows all common options supported by the Command Line Interface. For a description of the options, see [“Common Options” on page 193](#).

## Usage Notes for Exporting a Product Instance

- If you run the command without the products, instances, fixes, and configuration options, the generated template includes all the information about the product instance: instance properties, fixes, configurations, and inline templates for the products hosted on the instance and their configurations.

- When using `options=CONFIGURATION`, the command exports all configurations for the specified product instance, including the configurations for its child run-time components, if any. To select and extract only specific configurations, use the `sagcc exec templates composite generate input` command.
- When using `options=PRODUCTS` and `options=INSTANCES`, the generated template includes the following template snippet:

```
environments:
  default:
    repo.product: ${}          # Parameter for the product repository from
                                # which to install products
layers:
  product-template-layer:
    templates: product-instance-template
    productRepo: ${repo.product} # Placeholder parameter for the product repository
templates:
  product-instance-template:
    products:
      product1:
      product2:
      product3:
        instanceProperty1: instancePropertyValue1
        instanceProperty2: instancePropertyValue2
```

- When using `options=FIXES`, the generated template depends on the version of the source installation.

If the source installation is version 10.5 and higher, the command generates a template with fixes and patches parameters that list the fixes installed on the source installation. When applying the template snippet from the following example, Command Central will install the same fixes (and patches) as the ones installed on the source installation:

```
dslVersion: "1.1"          # Added only when exporting a list of patches.
environments:
  default:
    fixes: [fix1, fix2, fix3]
    patches: [patch1, patch2] # Supported only in DSL "1.1" and higher.
    repo.fix: ${}           # Parameter for the fix repository from
                            # which to install fixes.
layers:
  product-template-layer:
    templates: product-instance-template
    fixRepo: ${repo.fix}    # Placeholder parameter for the fix repository.
provision:
  default:
    product-template-layer: ${nodes}
templates:
  product-instance-template:
    fixes: ${fixes}        # A fixes section in the inline template of
                            # the product instance that refers to the
                            # "fixes" parameter.
    patches: ${patches}    # A patches section in the inline template
                            # of the product instance that refers to the
                            # "patches" parameter.
```

If the source installation is version 10.4 and lower, the command generates a template with the `fixes: ALL` parameter. When applying the template snippet from the following example, Command Central will install all fixes available for the product instance in the fix repository:

```
environments:
  default:
    prefix.fixes: ALL          # Parameter that indicates to install all fixes
                              # for the product instance.
    repo.fix: ${}             # Parameter for the fix repository from which
                              # to install fixes.
layers:
  product-template-layer:
    templates: product-instance-template
    fixRepo: ${repo.fix}      # Placeholder parameter for the fix repository.
provision:
  default:
    product-template-layer: ${nodes}
templates:
  product-instance-template:
    fixes: ${prefix.fixes} # A fixes section in the inline template of the
product
                              # instance that refers to the "prefix.fixes"
parameter.
```

- Command Central always extracts license keys and paths to file locations, which cannot be resolved from the Software AG installation directory, as parameters with required values, for example:

```
environments:
  default:
    instance.license.key: ${}
    instance.data.dir: ${}
```

You must provide the required values for such parameters when creating a layer using the layer definition of the generated template. If you apply the generated template with the `apply composite templates` command, specify the values in a properties file or as parameters in the `apply composite templates` command.

- When using `options=PRODUCTS`, the list of product inline templates in the generated template depends on the version of the source installation:
  - For installations version 10.4 or higher, the command adds inline templates only for the products hosted on the product instance.
  - For installations version 10.3 or lower, the command adds inline templates for all products from the source installation (not only the products hosted on the product instance).
- When generating the run-time micro template, Command Central does not extract files that the product instance uses.

## Usage Notes for Exporting a Platform Manager Instance

- When you specify only the `options=INFRASTRUCTURE` argument, Command Central exports all data for the Platform Manager instance. You can use `options=PRODUCTS`, `options=FIXES`, and

`options=CONFIGURATION` to determine what information about Platform Manager you want to include in the `templates:` section of the generated template.

- `options=PRODUCTS` Includes a list of the Platform Manager components and any installed product plugins in the `templates:templateAlias:products:` section.
- `options=FIXES` Includes a `templates:templateAlias:fixes:` section
- `options=CONFIGURATION` Includes all configuration instances of all configuration types for the Platform Manager instance. You can filter which configurations to include in the generated template using the `sagcc exec templates composite generate input` command.
- When exporting a Platform Manager instance with products, the generated template includes Platform Manager plugins only if the installation contains Platform Manager plugins that are not part of a runtime instance. For example, if the installation contains a Platform Manager plugin for Integration Server but no Integration Server runtime instances, then Command Central lists the plugin in the `products:` section of the infrastructure template. Command Central lists all other Platform Manager plugins in the runtime template for the respective product.
- After exporting a Platform Manager instance with fixes, you must check the version of the Command Central bootstrapper that you will specify in the `cc.installer: ${}` parameter in the generated template. If you specify a bootstrapper that contains fixes with a higher version than the ones listed in the generated template, the fixes from the bootstrapper will get installed and will replace the ones from the template.

### Usage Note for Exporting an Instance to a Migration Template

When you include the `options=INFRASTRUCTURE` argument, Command Central generates a migration template that you can use to migrate the product instance. The generated template includes a `nodes:` section with the infrastructure data of the Platform Manager node, on which the product instance is installed.

For example, to generate a migration template with alias "IS-default-migration" for the "OSGI-IS\_default" product instance, installed in the local Platform Manager installation:

```
sagcc exec templates composite generate alias=IS-default-migration
nodeAlias=local runtimeComponentId=OSGI-IS_default
options=INFRASTRUCTURE options=INSTANCES options=PRODUCTS
```

### Usage Note for Exporting a Database

Note that the generated database template does not include the database storage component. Before you apply the database template in an installation or use it to create a database layer in a product stack, you must ensure that the database storage component for the database is already created.

### Examples of Exporting a Product Instance to a Template

To extract all configurations for the "OSGI-IS\_default" run-time component, installed in the "local" installation, to a template with alias "is-config" and use the "is" prefix for the parameters in the

generated template. If a template with the specified alias exists, the generated template will overwrite it.

```
sagcc exec templates composite generate alias=is-config nodeAlias=local
parameterize=true
paramPrefix=is options=CONFIGURATION runtimeComponentId=OSGI-IS_default overwrite=true
```

To extract all instance properties and configurations for the "OSGI-IS\_default" product instance, installed in the "local" installation, to a template with alias "is-config".

```
sagcc exec templates composite generate alias=is-config nodeAlias=local
options=INSTANCES
options=CONFIGURATION runtimeComponentId=OSGI-IS_default
```

To generate a template with all the information about the "OSGI-IS\_default" product instance, installed in the "local" installation, to a template with alias "is-config".

```
sagcc exec templates composite generate alias=is-config nodeAlias=local
runtimeComponentId=OSGI-IS_default
```

## Examples of Exporting a Platform Manager Instance to a Template

To extract and parameterize all configurations for the local Platform Manager, to a template with alias "SPM-infra". If a template with the specified alias exists, the generated template will overwrite it.

```
sagcc exec templates composite generate alias=SPM-infra nodeAlias=local
parameterize=true options=INFRASTRUCTURE options=CONFIGURATION overwrite=true
```

## Example of Exporting a Database to a Template

To export the database components and schemas for the products in the installation with alias "sag1" to a database template with alias "db-template1", using the database connection details for the database with name "myDatabase" and type "sqlserver":

```
sagcc exec templates composite generate alias=db-template1
nodeAlias=sag1 options=DATABASE
db.url="jdbc:wm:Sqlserver://rubicon:1433;databaseName=myDatabase"
db.username=myDbUser db.password=myDbUser_password db.type=sqlserver overwrite=true
dbUseComponentsLatestVersion=false -s http://localhost:8090
```

The generated database template is automatically imported into the local Command Central installation and will overwrite any imported template with the same template alias. The database components and schemas will get exported with the version of the source installation.

## sagcc exec templates composite generate input

Generates a template using input parameters, specified in a template metadata file. Based on the template metadata file, the command can generate:

- A run-time micro template from an installed product instance.
- An infrastructure micro template from the Platform Manager instance in an installation.

- A product instance template containing a `nodes:` section.

The command can help you when you want to filter which configuration properties of the product instance and its child run-time components to extract in the template.

Command Central imports the generated template automatically and creates a layer type definition for the template in the local Command Central installation. You can use this layer type definition to create a layer for the product instance in an existing product stack. See the [“sagcc create stacks layers” on page 428](#) command.

## Syntax

- Command Central syntax:

```
sagcc exec templates composite generate [--input | -i] filename.xml [options]
```

- Not supported on Platform Manager.

## Arguments and Option

Argument or Option	Description
<code>{--input   -i}</code>	Required. Specifies a template metadata file in XML format. For information about the structure and contents of the input metadata file, see Usage Notes. For more information about the <code>{--input   -i}</code> option, see <a href="#">“input” on page 204</a> .
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## Usage Notes

- The input file with the template metadata for generating an infrastructure or a runtime template uses the following XML schema:
  - To specify that all configurations that are defined in the `<configurations>` element will be added in the generated template.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<generateCompositeTemplate>
  <alias>templateAlias</alias>
  <nodeAlias>exportInstallationAlias</nodeAlias>
  <runtimeComponentId>runtimeComponentId</runtimeComponentId>
  <infrastructure>
    <includeHostname>true</includeHostname>
  </infrastructure>
  <instances/>
  <products/>
  <fixes/>
  <configurations>
    <runtimeComponent id="runtimecomponentId1">
```

```

<configurationInstanceId>configurationInstanceId1</configurationInstanceId>
<configurationInstanceId>configurationInstanceId2</configurationInstanceId>
...
<configurationTypeId>configurationTypeId1</configurationTypeId>
<configurationTypeId>configurationTypeId2</configurationTypeId>
...
</runtimeComponent>
<runtimeComponent id="runtimeComponentId2">
  <configurationInstanceId>configurationInstanceId1</configurationInstanceId>
  ...
</runtimeComponent>
...
</configurations>
<parameterize>true</parameterize>
<paramPrefix>paramPrefix</paramPrefix>
<overwrite>true</overwrite>
</generateCompositeTemplate>

```

- To specify which configurations to exclude from the generated template, set `exclude="true"` attribute of the `<configurations>` element. All configurations for the specified run-time components will be added in the template, except the configurations listed in the `<configurations>` element of the template metadata file.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<generateCompositeTemplate>
  <alias>templateAlias</alias>
  <nodeAlias>exportInstallationAlias</nodeAlias>
  <runtimeComponentId>runtimeComponentId</runtimeComponentId>
  <infrastructure>
    <includeHostname>true</includeHostname>
  </infrastructure>
  <instances/>
  <products/>
  <fixes/>
  <configurations exclude="true">
    <runtimeComponent id="runtimeComponentId1">
      <configurationInstanceId>configurationInstanceId1</configurationInstanceId>
      <configurationInstanceId>configurationInstanceId2</configurationInstanceId>
      ...
      <configurationTypeId>configurationTypeId1</configurationTypeId>
      <configurationTypeId>configurationTypeId2</configurationTypeId>
      ...
    </runtimeComponent>
    <runtimeComponent id="runtimeComponentId2">
      <configurationInstanceId>configurationInstanceId1</configurationInstanceId>
      ...
    </runtimeComponent>
  </configurations>
  <parameterize>true</parameterize>
  <paramPrefix>paramPrefix</paramPrefix>
  <overwrite>true</overwrite>
</generateCompositeTemplate>

```

The following table describes the parameters in the template metadata file:

Parameter	Description
<code>templateAlias</code>	The alias name of the run-time micro template to generate.
<code>nodeAlias</code>	The alias name of the installation in which the parent product instance is installed.
<code>runtimeComponentId</code>	The ID of the parent product instance, from which to extract configurations. By default, if you do not include any child run-time components, only the configurations for the parent product instance will be extracted.
<code>infrastructure</code>	Required when the <code>&lt;runtimeComponentId&gt;</code> is "OSGI-SPM" or is not specified. If you omit the <code>&lt;infrastructure&gt;</code> element, the command generates a runtime micro template.
<code>includeHostname</code>	Optional. Includes the hostname of the Platform Manager node, from which you export Platform Manager.
<code>instances</code>	Includes the instance properties of the specified product instance.
<code>products</code>	Includes a list of product inline templates in the generated template. The list of products depends on the version of the source installation: <ul style="list-style-type: none"> <li>■ For installations version 10.4 or higher, the command adds inline templates only for the products hosted on the product instance.</li> <li>■ For installations version 10.3 or lower, the command adds inline templates for all products from the source installation (not only the products hosted on the product instance).</li> </ul>
<code>fixes</code>	Adds a <code>templates:templateAlias:fixes:</code> section. Only with DSL "1.1", if the source product instance has been updated with patches, it also adds a <code>templates:templateAlias:patches:</code> section. Based on the version of the source installation, the generated template will differ as follows: <ul style="list-style-type: none"> <li>■ 10.5 and higher, adds <code>fixes</code> and <code>patches</code> parameters that list the fixes and patches installed on the source product instance.</li> <li>■ 10.4 and lower, adds a <code>fixes: ALL</code> parameter that indicates to install all fixes available for the product instance in the fix repository.</li> </ul>
<code>configurations</code>	Includes the configurations of the product instance and all its child run-time components.

Parameter	Description
<code>exclude="true"</code>	<p>Optional. Specifies that the generated template will contain all configurations for the specified run-time components, except for the configurations listed in the template metadata file. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> - exclude the configurations listed in the <code>&lt;configurations&gt;</code> element</li> <li>■ <code>false</code> - (default) do not exclude the configurations</li> </ul>
<code>id="runtimeComponentId"</code>	<p>Optional. Specifies the IDs of the run-time components for which you want to export configurations. You can specify the ID of the parent run-time component, or one of its child components. You can list multiple run-time components.</p>
<code>configurationInstanceId</code>	<p>Optional. The ID of a specific configuration instance to be exported for the specified run-time component. You can list multiple configuration instances. If you do not specify any configuration instance IDs for a particular run-time component, all configurations for this run-time component will be exported.</p> <p><b>Note:</b> If <code>exclude="true"</code> the specified configuration instances will be excluded from the template.</p>
<code>configurationTypeId</code>	<p>Optional. Specifies that all configurations with this type ID must be included in the template. You can list multiple configuration type IDs.</p> <p><b>Note:</b> If <code>exclude="true"</code>, all configurations with the specified configuration types will be excluded from the template.</p>
<code>parameterize</code>	<p>Optional. Specifies whether to parameterize the exported configurations. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> - parameterize the configurations</li> <li>■ <code>false</code> - (default) do not parameterize the configurations</li> </ul>
<code>paramPrefix</code>	<p>Optional. Specifies a prefix to be added to the exported parameter names. Use this parameter only when <code>parameterize</code> is set to <code>true</code>.</p>
<code>createLayerType</code>	<p>Optional. Specifies whether to create a new layer type definition that references the generated template. The alias of the new layer type definition uses the alias of the generated</p>

Parameter	Description
	<p>template, for example if the template alias is <code>my-template</code>, the layer definition alias is <code>MY-TEMPLATE</code>. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> (default) - create a new layer definition</li> <li>■ <code>false</code> - do not create a new layer definition</li> </ul>
<code>overwrite</code>	<p>Optional. Use only when updating a template. Specifies whether to overwrite an existing template with the same alias name. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> - overwrite the template</li> <li>■ <code>false</code> - (default) do not overwrite the template</li> </ul>

### Example Metadata File for Exporting a Platform Manager Instance to an Infrastructure Template

Use the following template metadata file to generate an infrastructure micro template named "spm-infra," which contains the hostname of the Platform Manager node, where the "COMMON-MEMORY" configuration instance will be exported, configurations will be parameterized, and if a template with the same name already exists, it will be overwritten:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<generateCompositeTemplate>
  <alias>spm-infra</alias>
  <nodeAlias>local</nodeAlias>
  <runtimeComponentId>OSGI-SPM</runtimeComponentId>
  <parameterize>true</parameterize>
  <infrastructure>
    <includeHostname>true</includeHostname>
  </infrastructure>
  <configurations>
    <configurationTypeId>COMMON-MEMORY</configurationTypeId>
  </configurations>
  <overwrite>>false</overwrite>
</generateCompositeTemplate>
```

### Example Metadata File for Generating an Integration Server Template

Use the following template metadata file to generate a template named "is-config," which contains all typical Integration Server configurations from the "OSGI-IS\_default" Integration Server run-time component in the "local" installation, where configurations will be parameterized, the "is" prefix will be added in front of each exported parameter, and if a template with the same name already exists, it will be overwritten:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<generateCompositeTemplate>
  <alias>is-config</alias>
  <nodeAlias>local</nodeAlias>
  <runtimeComponentId>OSGI-IS_default</runtimeComponentId>
```

```

<configurations exclude="true">
  <runtimeComponent id="OSGI-IS_default">
    <configurationTypeId>COMMON-CONFIGURATION-TYPE-METADATA</configurationTypeId>
  </runtimeComponent>
  <runtimeComponent id="integrationServer-default">
    <configurationTypeId>COMMON-CONFIGURATION-TYPE-METADATA</configurationTypeId>
  </runtimeComponent>
</configurations>
<parameterize>true</parameterize>
<paramPrefix>is</paramPrefix>
<overwrite>true</overwrite>
</generateCompositeTemplate>

```

## Example Metadata File for Generating an Integration Server Template with a Nodes Section

Use the following template metadata file to generate a template named "is-config," which contains a nodes: section, all typical Integration Server configurations from the "OSGI-IS\_default" Integration Server run-time component in the "local" installation, where configurations will be parameterized, the "is" prefix will be added in front of each exported parameter, and if a template with the same name already exists, it will be overwritten:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<generateCompositeTemplate>
  <alias>is-config</alias>
  <nodeAlias>local</nodeAlias>
  <runtimeComponentId>OSGI-IS_default</runtimeComponentId>
  <infrastructure>
    <includeHostname>true</includeHostname>
  </infrastructure>
  <configurations exclude="true">
    <runtimeComponent id="OSGI-IS_default">
      <configurationTypeId>COMMON-CONFIGURATION-TYPE-METADATA</configurationTypeId>

    </runtimeComponent>
    <runtimeComponent id="integrationServer-default">
      <configurationTypeId>COMMON-CONFIGURATION-TYPE-METADATA</configurationTypeId>

    </runtimeComponent>
  </configurations>
  <parameterize>true</parameterize>
  <paramPrefix>is</paramPrefix>
  <overwrite>true</overwrite>
</generateCompositeTemplate>

```

## Example When Executing on Command Central

To generate a template using the properties, defined in the "my-is-config.xml" template metadata file:

```
sagcc exec templates composite generate -i my-is-config.xml
```

## sagcc exec templates composite generate migration

Generates a migration template that includes the infrastructure data of the source Platform Manager nodes. You can generate migration templates for one of the following supported migration types:

- Overinstall: same installation directory, hosts, and ports
- Side-by-side: new installation directory, same hosts and ports
- Cross-host: new hosts, same ports. The installation directory can have the same or a new value.

You can also export infrastructure data to a generic migration template that you can use for any of the supported migration types.

Command Central imports the generated template automatically and creates a layer type definition for the template in the local Command Central installation. You can use this layer type definition to create a layer in an existing product stack. See the [“sagcc create stacks layers” on page 428](#) command.

After you run the command, you can monitor the progress of the template generation job from the Jobs view in the Command Central web user interface.

### Syntax

- Command Central syntax:

```
sagcc exec templates composite generate alias=templateAlias
nodeAlias=nodeAlias options=migration
[migration.type={overinstall|sidebyside|crosshost|generic}]
[overwrite={true | false}] [options]
```

- Not supported on Platform Manager.

### Arguments and Options

Argument or Option	Description
<code>alias=<i>templateAlias</i></code>	Required. The alias name of the template to generate.
<code>nodeAlias=<i>nodeAlias</i></code>	Required. The alias name of the installation from which to generate a template.
<code>options=migration</code>	Required. Exports all data for the Platform Manager node to a migration template. The template includes data about the Platform Manager components, product plugins, fixes, configurations, and includes the parameters required for migrating the Platform Manager node. The migration parameters will depend on the migration type you set in the command.

Argument or Option	Description
<code>[migration.type={overinstall sidebyside crosshost generic}]</code>	<p>Optional. The type of migration for which to generate a template. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>overinstall</code></li> <li>■ <code>sidebyside</code></li> <li>■ <code>crosshost</code></li> <li>■ <code>generic</code> (default)</li> </ul> <p>When you do not include this argument, the command exports the data to a generic template.</p>
<code>[overwrite={true   false}]</code>	<p>Required when you execute the command to update a template. Specifies whether to overwrite an existing template with the same alias name. Valid values:</p> <ul style="list-style-type: none"> <li>■ <code>true</code> - overwrite the template</li> <li>■ <code>false</code> (default) - do not overwrite the template</li> </ul>
<code>[options]</code>	<p>Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a>.</p>

## Usage Notes

- You can generate a product instance template with a `nodes:` section and use it to migrate a product instance, as described in "Usage Notes for Exporting an Instance to a Migration Template" in [“sagcc exec templates composite generate” on page 457](#).
- When `migration.type=overinstall`, the Platform Manager connection and bootstrap details for the source nodes get exported to the `nodes:` section of the template. When you use the migration template for overinstall upgrade, the source node details in the `nodes:` section (such as installation directory, ports, and credentials) are also used for the target nodes.
- When `migration.type=sidebyside`, the Platform Manager connection and bootstrap details for the source nodes get exported to the `migration:nodes:` section of the template and have the `source.` prefix. When you use the migration template for side-by-side upgrade, the details in the `nodes:` section of the template specify the installation directory, ports, and credentials for the target nodes. Note that the `nodes.install.dir: ${}` is a required parameter, because you must always specify a new directory for the target installation when running a side-by-side migration.

Example:

```
dslVersion: ${dsl.version}
environments:
  default:
    cc.installer: ${} # The file name of the Command Central bootstrap installer
```

```

        # to bootstrap the target Platform Manager.
    dsl.version: '1.2' # The version of the template DSL.
    nodes.install.dir: ${} # The installation directory of the target Platform
Manager node.
    nodes.spm.credentials.key: ${} # The credentials alias for the target Platform
Manager.
    nodes.spm.is.secure.connection: true
    nodes.spm.port: ${} # The port of the target Platform Manager.
    nodes.ssh.login.credentials.key: ${} # The SSH login credentials alias for
the remote target machine.
    nodes.ssh.port: ${} # The SSH port of the remote target machine.
    nodes.ssh.substitute.credentials.key: ${nodes.ssh.login.credentials.key}
    source.nodes.install.dir: C:\sag\cc # The installation directory of the
source Platform Manager node.
    source.nodes.spm.credentials.key: ${} # The credentials alias to connect to
the source Platform Manager.
    source.nodes.spm.is.secure.connection: true
    source.nodes.spm.port: 8093 # The port of the source Platform Manager.
    source.nodes.ssh.login.credentials.key: ${} # The SSH login credentials
alias for the remote source machine.
    source.nodes.ssh.port: 22 # The SSH port of the remote source machine.
    source.nodes.ssh.substitute.credentials.key:
${source.nodes.ssh.login.credentials.key}
    ...
    migration: # The Platform Manager connection and bootstrap details for the source
nodes.
    nodes:
    default:
    default:
        port: ${source.nodes.spm.port}
        secure: ${source.nodes.spm.is.secure.connection}
        credentials: ${source.nodes.spm.credentials.key}
        bootstrapInfo:
            installDir: ${source.nodes.install.dir}
            port: ${source.nodes.ssh.port}
            credentials: ${source.nodes.ssh.login.credentials.key}
            substituteUserCredentials: ${source.nodes.ssh.substitute.credentials.key}

    nodes: # The Platform Manager connection and bootstrap details for the target
nodes.
    default:
    default:
        port: ${nodes.spm.port}
        secure: ${nodes.spm.is.secure.connection}
        credentials: ${nodes.spm.credentials.key}
        bootstrapInfo:
            installer: ${cc.installer}
            installDir: ${nodes.install.dir}
            port: ${nodes.ssh.port}
            credentials: ${nodes.ssh.login.credentials.key}
            substituteUserCredentials: ${nodes.ssh.substitute.credentials.key}

```

- When `migration.type=crosshost`, the Platform Manager connection and bootstrap details for the source nodes get exported to the `migration:nodes:` section of the template and have the `source.` prefix. When you use the migration template for cross-host upgrade, the details in the `nodes:` section of the template specify the host, installation directory, ports, and credentials for the target nodes. Note that the `nodes.local.host: ${}` is a required parameter, because

you must always specify a new host for the target installation when running a cross-host migration.

Example:

```
dslVersion: ${dsl.version}
environments:
  default:
    cc.installer: ${} # The file name of the Command Central bootstrap installer
                    # to bootstrap the target Platform Manager.
    dsl.version: '1.2' # The version of the template DSL.
    nodes.install.dir: ${} # The installation directory of the target Platform
    Manager node.
    nodes.local.host: ${} # The name of the host, on which to install the target
    Platform Manager node.
    nodes.spm.credentials.key: ${} # The credentials alias for the target Platform
    Manager.
    nodes.spm.is.secure.connection: true
    nodes.spm.port: ${} # The port of the target Platform Manager.
    nodes.ssh.login.credentials.key: ${} # The SSH login credentials alias for
    the remote target machine.
    nodes.ssh.port: ${} # The SSH port of the remote target machine.
    nodes.ssh.substitute.credentials.key: ${nodes.ssh.login.credentials.key}
    source.nodes.install.dir: C:\sag\cc # The installation directory of the
    source Platform Manager node.
    source.nodes.local.host: localhost # The name of the host, on which the
    source Platform Manager is installed.
    source.nodes.spm.credentials.key: ${} # The credentials alias to connect to
    the source Platform Manager.
    source.nodes.spm.is.secure.connection: true
    source.nodes.spm.port: 8093 # The port of the source Platform Manager.
    source.nodes.ssh.login.credentials.key: ${} # The SSH login credentials
    alias for the remote source machine.
    source.nodes.ssh.port: 22 # The SSH port of the remote source machine.
    source.nodes.ssh.substitute.credentials.key:
    ${source.nodes.ssh.login.credentials.key}
    ...
  migration: # The Platform Manager connection and bootstrap details for the source
  nodes.
  nodes:
    default:
      local:
        host: ${source.nodes.local.host}
      default:
        port: ${source.nodes.spm.port}
        secure: ${source.nodes.spm.is.secure.connection}
        credentials: ${source.nodes.spm.credentials.key}
        bootstrapInfo:
          installDir: ${source.nodes.install.dir}
          port: ${source.nodes.ssh.port}
          credentials: ${source.nodes.ssh.login.credentials.key}
          substituteUserCredentials: ${source.nodes.ssh.substitute.credentials.key}

    nodes: # The Platform Manager connection and bootstrap details for the target
    nodes.
    default:
      local:
        host: ${nodes.local.host}
      default:
        port: ${nodes.spm.port}
```

```
secure: ${nodes.spm.is.secure.connection}
credentials: ${nodes.spm.credentials.key}
bootstrapInfo:
  installer: ${cc.installer}
  installDir: ${nodes.install.dir}
  port: ${nodes.ssh.port}
  credentials: ${nodes.ssh.login.credentials.key}
  substituteUserCredentials: ${nodes.ssh.substitute.credentials.key}
```

- When `migration.type=generic`, the migration template includes both the `nodes:` and `migration:nodes:` section and the type of migration is determined by the values you specify for the parameters with the `nodes.` prefix. When applying a generic template, set the parameters for each migration type as follows:
  - **Overinstall:** set the same host in the `nodes.local.host:` and `source.nodes.local.host:` parameters, and the same installation directory in the `nodes.install.dir:` and `source.nodes.install.dir:` parameters.
  - **Side-by-side:** set the same host in the `nodes.local.host:` and `source.nodes.local.host:` parameters, but different installation directories in the `nodes.install.dir:` and `source.nodes.install.dir:` parameters.
  - **Cross-host:** set different hosts in the `nodes.local.host:` and `source.nodes.local.host:` parameters.

## Examples When Executing on Command Central

To export the infrastructure data for the local Platform Manager to a generic migration template with alias "local-migration-generic", which you can use for overinstall, side-by-side, or cross-host migration:

```
sagcc exec templates composite generate alias=local-migration-generic
nodeAlias=local options=migration
```

To export the infrastructure data for the local Platform Manager to a migration template with alias "local-migration-sidebyside", which you can use for side-by-side migration:

```
sagcc exec templates composite generate alias=local-migration-sidebyside
nodeAlias=local options=migration migration.type=sidebyside
```

## 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
<code>template_alias</code>	Required. The alias of the template to validate. You can determine the template alias using the <code>sagcc get templates composite</code> command.
<code>[templateParameter=value...]</code>	Optional. Specify template parameters that determine which validation checks the command will perform.
<code>[options]</code>	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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 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 129](#) 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 or a yaml file.

### Syntax

- Command Central syntax:

- To export a template into a zip archive:

```
sagcc get templates composite export templateAlias [--output | -o] filename.zip
[{--output-format | -f} application/vnd.sagcc.asset+zip] [options]
```

- To export a template into a yaml file:

```
sagcc get templates composite export templateAlias [--output | -o] filename.yaml
[{--output-format | -f} application/yaml] [options]
```

- Not supported on Platform Manager.

### Arguments and Options

Argument or Option	Description
<i>templateAlias</i>	Required. The alias of the template to export. You can determine the template alias using the <code>sagcc list templates composite</code> command.
<i>{--output   -o} filename.zip</i>	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 <a href="#">“output” on page 209</a> .
<i>{--output   -o} filename.yaml</i>	Required when exporting into a yaml file. The name of the output yaml file to which to export the template. For more information about the <code>{--output   -o}</code> option, see <a href="#">“output” on page 209</a> .
<i>[{--output-format   -f} application/vnd.sagcc.asset+zip]</i>	Optional. Use when exporting templates into composite assets. Indicates that the output file is a composite asset zip file, which contains an acdl file for the template and the template yaml file in a zip archive.
<i>{--output-format   -f} application/yaml</i>	Required when exporting into a yaml file. Indicates that the output file is a YAML template file.

Argument or Option	Description
[ <i>options</i> ]	The command allows all options supported by the Command Line Interface. For a description of the options, see “ <a href="#">Common Options</a> ” on page 193.

## Usage Notes

- When you include the `-f vnd.sagcc.asset+zip` option, the command exports the template as a Command Central composite asset zip file. You can add this zip archive to a local Software AG Common Landscape Asset Registry (LAR) and then deploy the template to a target on-premise installation. You must first unzip the asset zip file created with this command and then add the ACDL file and the template zip file into LAR. For more information about LAR, see *Software AG Infrastructure Administrator's Guide*. For more information about deploying assets, see *webMethods Deployer User's Guide*.
- The templates that you can export as composite assets are either templates created with the `sagcc exec templates composite generate` command, or any user-defined templates imported into Command Central.

## Examples 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
```

To export a template, under the alias “is-config” into a zip file, named “is-config.zip”, which contains an asset zip file and an acdl file for the template:

```
sagcc get templates composite export is-config -o is-config.zip
-f application/vnd.sagcc.asset+zip
```

To export a template, under the alias “is-config” into a yaml file, named “is-config.yaml”:

```
sagcc get templates composite export is-config -o is-config.yaml -f application/yaml
```

## 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
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

## 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
```

## sagcc list templates composite properties

Retrieves a list of all environment properties defined in a `environments:env.type:` section of a Command Central template.

### Syntax

- Command Central syntax:

```
sagcc list templates composite properties templateAlias [environment.type=type]
[options]
```

- Not supported on Platform Manager.

## Arguments and Options

Argument or Option	Description
<i>templateAlias</i>	Required. The alias of the template from which to retrieve properties.
[environment.type= <i>type</i> ]	Optional. The environment type for which to retrieve the environment properties. If you do not specify the environment type, the command retrieves the properties of the default environment.
[ <i>options</i> ]	Optional. The command allows all options supported by the Command Line Interface. For a description of the options, see <a href="#">“Common Options” on page 193</a> .

### Example When Executing on Command Central

To retrieve a list of all environment properties from the default environment defined in the template with alias "sag-spm-proxy" :

```
sagcc list templates composite properties sag-spm-proxy
```



# 14 Configuring Command Central and Platform Manager

---

■ Configuration Types for Command Central and Platform Manager OSGI .....	484
■ Configuration Types for Command Central and Platform Manager OSGI ENGINE .....	487
■ Run-time Monitoring States for OSGI-CCE and OSGI-SPM .....	491
■ Run-time Monitoring States for OSGI-SPM-ENGINE .....	492
■ Server System Properties .....	493
■ Using the Secure Socket Layer (SSL) Protocol and Certificates in Command Central .	499
■ Preparing to Replace the Default Keystore and Truststore .....	499
■ Update the SSL Connection Settings for the CLI .....	502
■ Considerations When Using Configuration Properties .....	503

## Configuration Types for Command Central and Platform Manager OSGI

The OSGI-CCE and OSGI-SPM run-time components support creating configuration instances of different configuration types. The following table describes the configuration types that the OSGI-CCE and OSGI-SPM run-time components support and what you can configure with each configuration type.

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 <a href="#">“COMMON-PORTS Usage Notes” on page 485.</a>
COMMON-PROXY	OSGI-CCE, OSGI-SPM	The proxy server settings if you must route server requests through a third party server. See also <a href="#">“COMMON-PROXY Usage Notes” on page 485.</a>
COMMON-SYSPROPS	OSGI-CCE, OSGI-SPM	DEPRECATED. Use COMMON-JAVASYSPROPS.

## COMMON-LDAP Usage Notes

To configure a connection to an LDAP directory, you can use [sag-cc-ldap](#) or export an existing LDAP configuration to a template. Note that the value of the location parameter for the truststore and keystore should be a valid URI starting with `file:///`, for example:

```
products:
  CCE:
    default:
      configuration:
        OSGI-CCE:
          COMMON-LDAP:
            COMMON-LDAP-sag:
              TruststoreDefinition:
                Type: JKS
                Location: file:///C:/SoftwareAG/common/conf/ldap_truststore.jks
```

## 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 JMX 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"> <li>■ true - enabled (default for all proxy configuration types, except COMMON-PROXY-ALL)</li> <li>■ false - disabled</li> </ul> <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 noHost
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   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 different configuration types. The following table describes the configuration types that the OSGI-CCE-ENGINE and OSGI-SPM-ENGINE run-time components support and what you can configure with each configuration type.

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 <a href="#">“sagcc get configuration data” on page 238</a> command to use when connecting to a repository or a remote machine. See also <a href="#">“COMMON-CREDENTIALS Usage Notes” on page 488</a> .
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 <a href="#">“COMMON-JAAS Usage Notes” on page 489</a> .
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 <a href="#">“COMMON-JAAS-REALMS Usage Notes” on page 489</a> .
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.

Configuration type	Supported on	Use to configure...
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	The internal unique ID of Platform Manager. Command Central manages unique IDs automatically.  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.

## 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 with version 10.7 and lower requires basic authentication and includes default username and password for the administrator user.
  - COMMON-CREDENTIALS-ADMINISTRATOR - used when a product or repository with version 10.11 and higher requires basic authentication. Note that this credentials alias does not include default user password for the administrator user. You must specify a strong administrator password by navigating to **Environments > ALL > Instances > CCE >**

**Command Central Server > Configuration > Credentials** in the Command Central web user interface and editing the ADMINISTRATOR credentials.

- 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 [Configuration Commands](#).

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 a configuration instance of a layer definition that you use when defining a layer in a product stack.

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.
- CCE-LAYER-TYPES-DATABASE-EXISTING  
Defines a database layer to use when you want to connect to an existing database.

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 key performance indicators (KPIs) used for monitoring. Take corrective actions when the KPI threshold approaches critical values. The following table lists the KPIs that are used and describes what they monitor.

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"> <li>■ Marginal is 80% CPU usage.</li> <li>■ Critical is 95% CPU usage.</li> <li>■ Maximum is 100% CPU usage.</li> </ul> <p>KPI limitations:</p> <ul style="list-style-type: none"> <li>■ supported only when running Java 7 or higher</li> <li>■ not reported when running on HP-UX</li> </ul>
JVM Memory	<p>How much JVM memory the run-time component uses.</p> <p>KPI values:</p> <ul style="list-style-type: none"> <li>■ Marginal is calculates as follows:  <math>\text{MAX}(\text{Maximum} * 80\%, \text{Maximum} - 100)</math>            A marginal value is when only 20% of the JVM memory is free or less than 100MB of JVM memory is available.</li> <li>■ Critical is calculated as follows:  <math>\text{MAX}(\text{Maximum} * 95\%, \text{Maximum} - 50)</math>            A critical value is when only 5% of the JVM memory is free or less than 50MB of JVM memory is available.</li> <li>■ Maximum is the amount of memory allocated to the JVM.</li> </ul> <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"> <li>■ Marginal and Critical are the current number of the allocated JVM threads plus one thread: <math>\text{current} + 1</math>.</li> </ul>

KPI	Use to monitor...
	<ul style="list-style-type: none"> <li>■ Maximum is the maximum allocated JVM threads plus one thread: MAX+1</li> </ul>

## 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 key performance indicators (KPIs) used for monitoring. Take corrective actions when the KPI threshold approaches critical values. The following table lists the KPIs that are used and describes what they monitor.

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"> <li>■ Marginal is 80% CPU usage.</li> <li>■ Critical is 95% CPU usage.</li> <li>■ Maximum is 100% CPU usage.</li> </ul>
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"> <li>■ Marginal is 80% of the maximum amount of physical disk space.</li> <li>■ Critical is 95% of the maximum amount of physical disk space.</li> <li>■ Maximum is the total amount of physical disk space.</li> </ul>
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"> <li>■ Marginal is 80% of the maximum amount of physical memory.</li> <li>■ Critical is 95% of the maximum amount of physical memory.</li> <li>■ Maximum is the total amount of physical memory.</li> </ul>

## 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. For a sample template, see [sag-cc-tuneup](#)

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

### **com.softwareag.platform.management.client.template.composite.migration.registry.cleanup.seconds**

The time interval in seconds after which Command Central cleans up the migration registry. When a template is executed successfully, Command Central automatically cleans up the registry after 60 hours.

Default: 216000

## 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.remoteaccess.keepaliveperiodsec**

The time interval in seconds between keep-alive messages from the client to the remote SSH server. Use this property to keep the SSH connection open when Command Central runs long-running operations on a remote machine and the SSH server is configured to terminate idle sessions. To send keep-alive messages, set a positive integer greater than zero.

Default: 0 - Indicates not to send keep-alive messages.

You must set a keep-alive interval that is smaller than the terminate idle sessions interval, configured on the remote SSH server. For example, if the SSH server terminates idle sessions after 180 seconds, set the keep-alive property to 150 seconds.

**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.checkonline.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 with the IDs of the configuration instances that you must specify on one line. Do not specify the same property with a different value on separate lines.

**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.

You must specify the values for this property on one line. Do not specify the same property with a different value on separate lines.

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.client.template.composite.skip.restart.for`**

Indicates whether to bypass the restart of specific run-time components at the end of the composite template application. The value is a comma-separated list with the IDs of the run-time components that you must specify on one line. Do not specify the same property with a different value on separate lines. For example, to skip the restart of the run-time component with IDs "TES-default", "IS-instance1", "IS-instance2":

```
com.softwareag.platform.management.client.template.composite.skip.restart.for=TES-default,  
IS-instance1,IS-instance2
```

Restart Command Central after setting the property, and then apply the template.

### **`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

## Repositories

**com.softwareag.platform.management.client.fix.mirror.platform.validation**

Configure for OSGI-CCE. When creating a fix mirror repository, the property indicates whether to check if the source fix repository includes fixes for all platforms. To disable validating the platforms in the source fix mirror repositories:

```
com.softwareag.platform.management.client.fix.mirror.platform.validation=false
```

Default: true

## SSH Connections

**com.softwareag.platform.management.client.remoteaccess.allowedkexmethods**

Configure for OSGI-CCE. Controls the list and priority of key exchange methods allowed by Command Central for securing remote SSH connections. To modify the default list of key exchange methods and their priority, specify a comma-separated list with the key exchange methods that you want to keep enabled in the required priority order from highest to lowest. Command Central will disable the key exchange methods that are not on the specified list. For example, to enable only the diffie-hellman-group16-sha512 and the diffie-hellman-group18-sha512 key exchange methods and set diffie-hellman-group16-sha512 with a higher priority than diffie-hellman-group18-sha512:

```
com.softwareag.platform.management.client.remoteaccess.allowedkexmethods=diffie-hellman-group16-sha512,
diffie-hellman-group18-sha512
```

The following key exchange methods are allowed, but disabled by default, because they are weak and should be enabled only when required by the environment:

- diffie-hellman-group1-sha1

#### ■ diffie-hellman-group-exchange-sha1

If you do not specify a list of key exchange methods in the property, Command Central allows only the key exchange methods that are enabled by default in the preset order of priority. Default list (in order of priority, from highest to lowest): `diffie-hellman-group14-sha256`, `diffie-hellman-group16-sha512`, `diffie-hellman-group18-sha512`, `diffie-hellman-group14-sha1`

## Disk Compaction Strategy for the Command Central Cache

### **com.tc.frs.compactor.policy**

Configure for OSGI-CCE. Specifies the disk compaction strategy that Command Central uses when storing files in the *Software AG\_directory* `/profiles/CCE/data/cache/persistence` directory. By default, this property is set to `LSNGapCompactionPolicy`, which favors performance to disk storage requirements. However, when required, you can update the disk compaction policy to size-based, which favors predictable disk usage to performance, by setting:

```
com.tc.frs.compactor.policy=SizeBasedCompactionPolicy
```

Default: `LSNGapCompactionPolicy`

### **com.tc.frs.io.nio.segmentSize**

Configure for OSGI-CCE. Specify only when `com.tc.frs.compactor.policy=SizeBasedCompactionPolicy`. Controls the size of the FRS files in bytes. The default size is 1 MB (1048576 bytes). In the example above the segment size is set to 100 MB (100 x 1024 x 1024). If you update the default value, keep in mind that when you reduce the maximum size of the FRS files, the system will create more files, which might lead to more wasted disk space.

For example, to set the size of the FRS files to 100 MB (100 x 1024 x 1024 bytes):

```
com.tc.frs.io.nio.segmentSize=104857600
```

Default: 1048576 (equivalent to 1 MB)

### **com.tc.frs.compactor.sizeBased.threshold**

Configure for OSGI-CCE. Specify only when `com.tc.frs.compactor.policy=SizeBasedCompactionPolicy`. Sets when to start the on-disk data compaction process. Specify the threshold for starting the disk compaction as a decimal number, representing the percentage of disk space that the in-memory data is using. For example, to set the compaction process to start when the in-memory data is using 50% of the disk space:

```
com.tc.frs.compactor.sizeBased.threshold=0.50
```

Default: 0.50

## com.tc.frs.compactor.sizeBased.amount

Configure for OSGI-CCE. Specify only when

`com.tc.frs.compactor.policy=SizeBasedCompactionPolicy`. Controls the amount of data to be compacted at a time. Specify the amount of data as a decimal number, representing a percentage of the total amount of data. For example, to specify that 5% of the data should be compacted at a time:

```
com.tc.frs.compactor.sizeBased.amount=0.05
```

Default: 0.05

## Using the Secure Socket Layer (SSL) Protocol and Certificates in Command Central

Software AG recommends disabling the HTTP port in a production environment, because it does not provide protection when sending sensitive information over the network. You can configure Command Central to communicate over SSL (using an HTTPs port) to ensure a secure session when communicating with the Platform Manager server, the CLI client, and browsers.

The default HTTPs ports of Command Central and Platform Manager point to the default keystore and truststore that are available after installing Command Central. The default keystore and truststore files contain keys and certificates that prove the identity of the Command Central server, Platform Manager server, and the CLI client. They also encrypt sensitive data sent during the secure session. Software AG strongly recommends that you replace the default keystore and truststore files with custom ones, created specifically for your organization.

## Preparing to Replace the Default Keystore and Truststore

Use a key and certificate management tool to generate the custom keys and certificates for Command Central. For example, you can use `keytool` and follow the instructions in [How do I generate keystores and certificates for Command Central](#) to generate the key and certificate files. For details about `keytool`, see the Java SE documentation in the Oracle Help Center. Make sure that you generate and store the certificates in a secure directory. Command Central does not have any special naming requirements for the file names of the generated keys and certificates. The following are some file naming suggestions that you could use:

- For the Command Central server keys/certificates, you can use the server hostname or IP address.
- For the Platform Manager keys/certificates, you can use the node alias.
- For the client truststores, you can choose any meaningful name.

To generate self-signed certificates, with the certificate management tool create the following files:

- The certified authority (CA) root key, for example `ccroot.jks`
- The CA certificate to import into a truststore or in a browser CA list, for example `ccroot.cer`

- A server key for the Command Central server and for each Platform Manager node that Command Central manages, for example `ccnode.jks` and `spmnode.jks`

Import the generated certificates into the client truststores:

- For the Command Central web user interface, import `ccroot.cer` and `spm*.cer` (that is the certificates for each Platform Manager node managed by Command Central) into the `cce-truststore.jks`.
- For the Command Central CLI, import `ccroot.cer` and `cc*.cer` into the `cli-truststore.jks`.

Copy the keystore and truststore files to a secure directory with controlled user access:

- On the machine that hosts Command Central (and the local Platform Manager)
- On each machine with a Platform Manager installation that Command Central manages

## Configuring the HTTPs Ports to Use the Custom Keystore

Before you start configuring the HTTPs ports to use the custom keystore, switch the communication between Command Central and all Platform Managers (local and remote) to HTTP.

Following is the recommended order, in which to configure the HTTPs ports to use the custom keystore and truststore:

1. Update the HTTPs port of the Command Central server.
2. Update the HTTPs port of the local Platform Manager installation.
3. Update the HTTPs port of each remote Platform Manager installation managed by Command Central.

You can watch the [Configure SSL in Command Central](#) video for a step-by-step demonstration.

## Update the HTTPs Port of the Command Central Server

You can update the default HTTPs port of Command Central to use the custom keystore, or configure a new HTTPs port (with unique alias and port number). However, if you choose to create and enable a new HTTPs port, you should disable the default HTTPs port of Command Central to avoid confusion.

In the HTTPs port configuration for CCE (in the web user interface; or `OSGI-CCE` if you use the `sagcc create configuration data` CLI command), specify the following security configuration details:

- **Key Alias** - An alias for the custom keystore.
- **Type** - The keystore type. Command Central supports the JKS and PKCS12 keystore types. The default is JKS.
- **Location** - The location of the custom keystore file, for example `/path/to/ccnode.jks`.
- **Password** - The password to access the custom keystore.

To verify the HTTPs connection to the Command Central server, use the "https://" URL of Command Central in a browser. For example, if you updated the default HTTPs port to use the custom keystore, type "https://localhost:8091/cce". Note that the browser might still indicate (with a warning or an error message) that the connection is not secure, because the browser does not recognize your CA certificate as issued by a verified authority. To get the secure connection green mark from the browser, you should import the custom CA certificate you generated (for example `ccroot.cer`) in the browser CA list.

## Update the HTTPs Port of Platform Manager

You can update the default HTTPs port of Platform Manager to use the custom keystore, or configure a new HTTPs port (with unique alias and port number). However, if you choose to create and enable a new HTTPs port, you should disable the default HTTPs port of Platform Manager to avoid confusion.

### ➤ To set up SSL communication between the local Platform Manager and Command Central

1. In the HTTPs port configuration for SPM (in the web user interface; or `OSGI-SPM` if you use the `sagcc create configuration data` CLI command), specify the following security configuration details:
  - **Key Alias** - An alias for the custom keystore.
  - **Type** - The keystore type. Command Central supports the JKS and PKCS12 keystore types. The default is JKS.
  - **Location** - The location of the custom keystore file, for example `/path/to/spmnode.jks`.
  - **Password** - The password to access the custom keystore.
2. Set the SSL connection settings of the Command Central server to point to the custom truststore:
  - a. In the Command Central web user interface go to, **Home > Instances > Environments > CCE > Command Central Server**.
  - b. On the Configuration tab, select **General Properties** and click **Outbound SSL Connection Settings**.
  - c. Update the truststore location to point to the custom truststore file, for example `/path/to/cce-truststore.jks` and specify the password to access the custom truststore.
  - d. Clear the **Ignore host verification** checkbox.

To test the SSL connection, go to the **Local** installation and on the Overview tab make sure that the **Port** field has the port number of the updated HTTPs port and **Use SSL** is selected. Then verify if the status of the **Local** installation is green.

After successfully testing the SSL communication with the local Platform Manager node, you can update the HTTPs port of each remote Platform Manager node managed by Command Central,

and test that the remote Platform Manager communicates with Command Central using the custom keystore and truststore.

## Update the SSL Connection Settings for the CLI

Command Central comes with a default configuration properties file that contains the SSL properties with the default SSL settings. The default `cc.properties` file is used by the Command Central server to communicate with the CLI over HTTPS. The following table lists the properties in the file, their default values and the command option that you can specify when running a CLI command to override the value specified in the properties file:

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

### 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. You must create a custom `cc.properties` file when you want to update the SSL connection settings for the CLI.

### ➤ To create a custom `cc.properties` configuration file

1. Go to `SoftwareAG_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. Open the custom `cc.properties` file in a text editor to update the SSL properties. Add `@secure` in front of a property name to indicate that the property value should be encrypted. For example: `@secure.password=manage`

```
# in the HTTPs URL of Command Central, set the port number of the HTTPs
port that points to the custom keystore
server=https://localhost:8091/cce

# the user name of the Command Central administrator user
username=Administrator

# the password of the Command Central administrator user
@secure.password=admin_password

# the location of the custom truststore file
ssl-truststore-file=path/to/cli-truststore.jks

# the password to access the truststore file
ssl-truststore-password=strong_password

# indicates that the CLI will only trust the Command Central server host
ssl-trust-all-hosts=false
```

5. Save the custom `cc.properties` file.
6. Run a CLI command with `--configuration-file path/to/custom.cc.properties` to set the CLI to use the SSL connection settings in the custom `cc.properties` file.

After the CLI command that includes the custom `cc.properties` file gets executed, Command Central encrypts the values of the properties prefixed with `@secure`.

## Considerations When Using Configuration Properties

### Determining the Value for Configuration Properties

When executing a CLI command, Command Central determines the value for any of the configuration properties in the following order:

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



# 15 Introduction to Command Central REST API

---

■ About Command Central REST API .....	506
■ Securing Command Central REST API .....	506
■ Command Central REST API Resources .....	506
■ Supported Media Types .....	507
■ HTTP Response Codes .....	507
■ Summary of REST Services .....	508

## 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 508](#).

- 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/landscape/nodes
```

```
curl -u Administrator:manage -X GET -H "Accept: application/xml"
http://localhost:8090/cce/landscape/nodes
```

## 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 508](#).

## HTTP Response Codes

The Command Central REST API response returns an HTTP response code that indicates success or error of the requested operation. The following table describes the HTTP response codes.

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 508](#).

## 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.
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 uninstall products and fixes.  PREVIEW FEATURE. Installs and uninstalls assets.
Repository	Manages product, fix, and assets repositories.

Service Name	Description
Security	Manages security credentials.
Stacks	Creates stacks and layers.
Template	Manages templates of environments and installations.



# 16 Logging and Troubleshooting

---

■ Using the Command Central and Platform Manager Logs .....	512
■ Activity Appears to Stop During File Download .....	520
■ Bootstrapper Fails with Package Error Such as "installer.jar not found" .....	520
■ Expected Product-specific Features are Not Available .....	520
■ Cannot Connect to Repositories Due to Invalid Credentials .....	520
■ Cannot Create Mirror Repository Because of Network Issues .....	520
■ Command Central Cannot Connect to Platform Managers .....	521
■ Values with Special Characters in a Template YAML File .....	521
■ Collecting Diagnostic Information with the Syscap UNIX Shell Script .....	521
■ Applying a Template Fails Because Product Configurations Fail to Apply .....	522
■ Setting the Values of Java System Properties in the Command Central Web User Interface and CLI .....	522
■ Remote Operations Run by Command Central over SSH get Terminated .....	522
■ Command Central Cache Takes too Much Disk Space .....	522

## Using the Command Central and Platform Manager Logs

The topics on this page describe the Command Central and Platform Manager Logs and the steps to follow when you want to manage the logs.

### Viewing Logs in Command Central

In the Command Central web user interface, go to **Environments > ALL**, click the instance or component whose logs to view, and click the Logs tab.

The following table explains how to download one, multiple, or all logs.

Logs to Download Steps	
One	Click  for that log.
Multiple	Hold down the Shift or Ctrl key and select the log rows. Click  and then click <b>Download selected logs</b> .
All	Click <b>Download selected logs</b> without selecting any log.

### 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.

Note that the logs might include user details required for auditing or troubleshooting, for example external user IDs. For information about how to delete the logs, see [“Deleting Logs” on page 518](#).

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 516](#).

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 > csrfpreventionontoken: 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 &lt; 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`.

Note that the logs might include user details required for auditing or troubleshooting, for example external user IDs. For information about how to delete the logs, see [“Deleting Logs” on page 518](#).

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 516](#).

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 512](#).

## 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.

## LAR Log

The `lar.log` contains messages that are generated by the Landscape Asset Repository (LAR).

## 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 517](#).

## 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 log4j2.properties -o
log.properties
```

- For the Platform Manager logs:

```
sagcc get configuration data node_alias OSGI-SPM log4j2.properties -o
log.properties
```

The log.properties file from the output of the command contains the log configuration settings.

2. Open the log.properties file in a text editor and change the values of the parameters or add a new logger 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 log4j2.properties -i
log.properties
```

- For the Platform Manager logs:

```
sagcc update configuration data node_alias OSGI-SPM log4j2.properties -i
log.properties
```

## 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 348 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 517.

## Activity Appears to Stop During File Download

---

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 Package Error Such as "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 using those credentials.

- 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**. If the product license agreement appears, the agreement has not yet been accepted for these credentials. Read the license text and, if you agree, accept the agreement now. Then return to Command Central and try again to connect to the Software AG repositories.

## 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.

## Values with Special Characters in a Template YAML File

Specifying passwords or other values that contain special characters in the template YAML file might result in YAML syntax errors. To resolve these errors, see the YAML specification about how to deal with special characters.

For passwords that include special characters, you can encrypt the password using the `“sagcc exec security encrypt”` on page 420 command.

## Collecting Diagnostic Information with the Syscap UNIX Shell Script

The `syscap` script is a UNIX shell script that you can use for capturing system information that can help when troubleshooting issues. The script collects technical information and generates a file with that information. It does not collect any user-related information. Software AG recommends sending the output file from the script to Software AG Global Support when you report issues.

The source code of the script is open for review. Before bootstrapping Platform Manager installations on a new UNIX host, you can copy the script to the target host and use it to troubleshoot issues during the bootstrapping operation.

You should run the script either as root (recommended), or as the user that has administrator privileges for Platform Manager. If you run the script as root, you must include the `-u SPM_admin_user` option in the command, where `SPM_admin_user` is the Platform Manager administrator user. If you run the script as the Platform Manager administrator user, do not include the `-u SPM_admin_user` option in the command.

To run the `syscap` script:

1. Open a UNIX shell window and type the following:

```
cd SoftwareAG_directory/PlatformManager/diagnostics
./syscap -u SPM_admin_user
```

2. Find the output file, named "syscap.txt.bz2" in the current working directory of the user.

For example: `SoftwareAG_directory/PlatformManager/diagnostics/syscap.txt.bz2`

## Applying a Template Fails Because Product Configurations Fail to Apply

---

Check the details of the template apply job that reported the ERROR status in the Jobs View of the Command Central web user interface or by running the [“sagcc list jobmanager jobs” on page 303](#) command. The error message in the Status Description indicates the alias of the failed template and details about the failed product configuration, such as run-time component ID, configuration instance ID, and configuration type. You can also check the Command Central and Platform Manager default logs for details about the product configuration failure.

If the configuration instance is not defined correctly, modify the configuration instance details in the `template.yaml` file, re-import the updated template in Command Central, and re-try applying the template.

## Setting the Values of Java System Properties in the Command Central Web User Interface and CLI

---

When setting the value of a Java system property in the Command Central web user interface or CLI, you must enclose values that contain spaces in quotes (""), and you must escape some characters in the value (such as backslashes and leading whitespaces) to get the value processed correctly. For details about which characters to escape in a property value, see the Java documentation.

## Remote Operations Run by Command Central over SSH get Terminated

---

When an SSH server is configured to terminate idle sessions and Command Central runs a long-running operation on a remote machine, the SSH connection might terminate before the operation is completed. To keep the SSH connection open, you can set the following Java system property:

```
com.softwareag.platform.management.client.remoteaccess.keepaliveperiodsec=<time-interval-in-seconds>
```

The keep-alive interval should be smaller than the terminate idle sessions interval, configured on the remote SSH server. For more information about the keep-alive property, see [“Provisioning” on page 494](#).

## Command Central Cache Takes too Much Disk Space

---

In case you have to reduce the size of the Command Central cache, stored in the `Software AG_directory /profiles/CCE/data/cache/persistence` directory, you can change the default disk compaction strategy used by Command Central. By default, Command Central uses a compaction strategy that favors performance to disk storage requirements. However, when required, you can update the disk compaction policy to size-based, which favors predictable disk usage to performance.

To update the disk compaction policy, on the Command Central web user interface, navigate to **Environments > ALL > Instances > CCE > Configuration > Java System Properties**, click **Edit**, and add the following lines:

```
com.tc.frs.compactor.policy=SizeBasedCompactionPolicy
com.tc.frs.io.nio.segmentSize=104857600
com.tc.frs.compactor.sizeBased.threshold=0.50
com.tc.frs.compactor.sizeBased.amount=0.05
```

Note that the system has to work for some time after updating the disk compaction policy, before it starts reducing the cache. For more information about:

- The disk compaction properties, see [“Disk Compaction Strategy for the Command Central Cache” on page 498](#).
- The FRS disk compaction policy, see the BigMemory Max documentation.

