



OPENCM USER GUIDE

WEBMETHODS CONFIGURATION MANAGEMENT

Version 3.0.0 | 02.12.2020

VERSION

Version	Date	Author	Description
3.0.0	02.12.2020	Håkan Hansson	Complete overhaul of user interface and implementation.

ABOUT SOFTWARE AG

Software AG offers the world's first Digital Business Platform. Recognized as a leader by the industry's top analyst firms, Software AG helps you combine existing systems on premises and in the cloud into a single platform to optimize your business and delight your customers. With Software AG, you can rapidly build and deploy Digital Business Applications to exploit real-time market opportunities. Get maximum value from big data, make better decisions with streaming analytics, achieve more with the Internet of Things, and respond faster to shifting regulations and threats with intelligent governance, risk and compliance. The world's top brands trust Software AG to help them rapidly innovate, differentiate and win in the digital world. Learn more at www.SoftwareAG.com.

© Software AG. All rights reserved. Software AG and all Software AG products are either trademarks or registered trademarks of Software AG. Other product and company names mentioned herein may be the trademarks of their respective owners

TABLE OF CONTENTS

1.	Introduction	5
1.1.	Overview	5
1.2.	Audience	5
1.3.	Terms & Abbreviations	5
1.4.	Summary Current Features	5
2.	Introducing a CM Strategy	7
2.1.	Configuration Repository	7
2.1.1.	None	7
2.1.2.	Static Repositories	7
2.1.3.	Repositories vs Automation	8
3.	Installation and Prerequisites	10
3.1.	Summary	10
3.2.	Compatibility	10
3.3.	Download	10
3.4.	Dependencies & Requirements	10
3.5.	Installation	10
4.	Using OpenCM	11
4.1.	Inventory	11
4.1.1.	Searching the Inventory	12
4.1.2.	External Connections	12
4.2.	Repository	13
4.3.	Anonymous Access	14
4.4.	Extractions	15
4.5.	Auditing	17
4.5.1.	Runtime Audits	17
4.5.2.	Governance Audits	18
4.6.	Command Central Node Definitions	20
5.	Managing OpenCM	22
5.1.	Managing the Inventory	22
5.1.1.	Defining an Installation	22

5.2.	Secrets	23
5.3.	Email Configuration	24
5.4.	Repository Synchronization	25

1. Introduction

1.1. Overview

The objective of this document is to describe the implementation of the OpenCM IS package and its context in which it operates. The OpenCM implementation is an open source project, located on <https://github.com/SoftwareAG/OpenCM>. Contributions to the code base are highly encouraged and welcomed.

The practice of Configuration Management is in this document covering the notion of storing, managing and operating on top of an off-line repository, which is the storage repository of how an environment, server or product component should be, and currently is, configured. For that reason, the CM practice is optimally involved during the whole lifecycle of the integration platform components: blueprinting, provisioning and configuration, quality assurance, change management, operation and maintenance, upgrades, decommissioning, etc.

1.2. Audience

This document is primarily intended for stakeholders responsible of managing the OpenCM component itself (developer, user) but also others who have an interest in the overall governance procedures of configuration properties management (of webMethods components).

The first part of the document provides a high-level overview of the CM strategy to provide a context for where OpenCM fits in. The latter half of the document provides step by step instructions on how to install, configure and use OpenCM.

1.3. Terms & Abbreviations

- **Configuration Management.** Abbreviated in this document as “CM” and refers to the overall practise of managing configurable information.
- **Inventory.** This is the base definition of servers, installations, port information, users and passwords.
- **Repository.** This is the storage area within OpenCM that reflects the runtime installation information. This storage area is meant to be continuously refreshed by extracted data.
- **Configuration Item.** Abbreviated as “CI” and not be confused with “Continuous Integration”. CI’s refer to individual properties that are defined within the CM repository.

1.4. Summary Current Features

OpenCM currently supports the following main features:

- **Inventory navigation:** through its user interface, it is possible to identify and locate information regarding a particular installation. For example, based on a server name, it is possible to quickly identify what it is used for.
- **Installation navigation of configuration properties:** once an installation has been extracted, all the configuration properties are available via the user interface. For example, it is possible to inspect the value of an Integration Server extended setting without having to use the regular IS Administrator UI.
- **Comparing multiple installations** – Ability to compare any configuration property between different installations. For example, it is possible to compare fix levels of multiple installations and report differences.

- **Establish a source-of-truth:** certain properties can be explicitly defined and audited to ensure conformance. For example, all Integration Servers must have an extended settings “watt.net.timeout” value of “300” – and a report can be generated to show which ones are not properly configured.
- **Command Central – Defining Groups and Installation Nodes:** as a helper function, OpenCM can generate the necessary CLI commands for Command Central that includes group information as well as custom passwords.

2. Introducing a CM Strategy

The primary reason for defining and implementing a CM strategy comes from the fact that business service quality is degraded without one. If the configuration properties of the runtime environments are not adequately documented, nor ensured, the unavoidable consequence is (undeliberate) differences or incorrectness in the setup of the platform within the various installations.

The objective of a CM strategy is to define an implementation approach that avoids the following (common) pitfalls:

- There are often/always discrepancies in terms of configuration settings between different runtime nodes and between different environments (where they actually should have been equal)
- There are often discrepancies between runtime environments and documentation
- The configuration information (what should be) is stale or completely absent
- Manual environment audits are time-consuming and cumbersome
- The same information is stored in multiple locations, thus needs to be updated in multiple places
- No clear understanding of wrong vs. right configuration property: i.e. there is no single source of truth
- Activities around installations, configurations and deployments are time consuming and error prone
- Quality Assurance is invalidated due to misconfigured environments

2.1. Configuration Repository

Central to the CM strategy is governance of configuration information and the level of control is then dictated by the ability to manage CI items in a separate location (from the runtime itself). What type of configuration repository to use is discussed here:

2.1.1. None

Not all organisations document the properties or settings that have been defined, updated and applied to their webMethods installations. The obvious consequence is that it is not possible to know what configurations are correct or incorrect. Whatever is defined in the runtime environment is the “source of truth”. A loss of server information (deliberate or undeliberate) will be hard to reproduce and the ability to provision new environments is equally difficult.

Inspection of a configuration setting in an environment and ensuring that it is correctly defined can only be done by comparing with another environment and by applying some level of subjective sanity check. Even then, it is not possible to know what the value of the property really should be.

2.1.2. Static Repositories

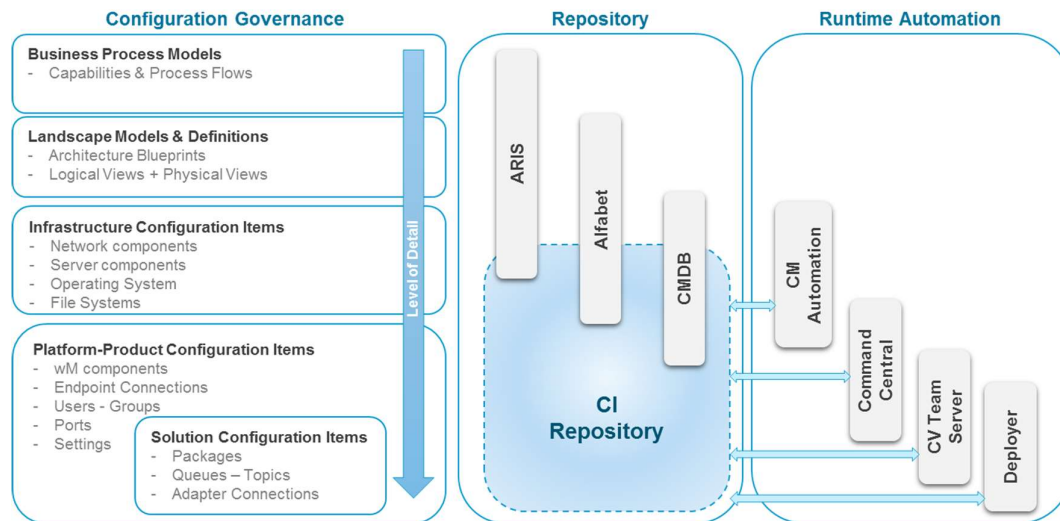
Somewhat more mature organisations will document the settings, the configurations and the properties of the installed environments and will most likely use a more static repository such as Word documents, Visio diagrams, Excel spreadsheets or a Wiki-type location. The problems with static repositories are:

- The repository will inevitably become out of synch with the environment over time. Even the most rigorous governance process will still prompt a manual, time-consuming environment audit to ensure that there are no misconfigurations.

- It is difficult to represent the environment using static repositories for different purposes. Architects, operations team, maintenance team, developers, change management, etc. are all “users” of the repository and each team wants to view the information from different angles. This often leads to having the same configuration properties/items defined in multiple locations, after having been extracted out and copied to other places. A change of configuration then leads to having to perform updates in multiple locations (which again leads to discrepancies between multiple repositories).

2.1.3. Repositories vs Automation

Within the CM practice, it is important to distinguish between runtime automation tools (which are often classified as “Configuration Management tools”) and the support systems that can operate on top of a store that holds configuration information.



- Illustrated on the left side of above diagram, shows the level of granularity for “configuration governance”. On a very high level, the business functionality is defined as a set of process models, thus capturing the underlying IT requirements. The implementation of the business requirements are then supported by various architectural views and prompts an infrastructure setup with webMethods installations and solution deployments.
- On the right side of the diagram, there are a number of different automation tools:
 - CM Automation is a collective group of commercial/open source tools such as Puppet, Chef, SaltStack, Ansible, etc.
 - Command Central is the SoftwareAG tool for performing installations, applying fixes and managing of various configuration settings in the runtime environment.
 - Cross Vista team Server and SoftwareAG Deployer are tools for automating the deployment efforts of individual solution implementations.
- In the middle of the diagram, are identified (potential) repositories of configuration information:
 - SoftwareAG ARIS provides a repository for the blueprints and the physical architectural views can be extended to cover finer granularity of the assets. However, the volume of data that is stored in a CM implementation is far too great. It is estimated that between 500-1000 properties per server installation

would need to be captured. The ability to model and define configuration items is inherent but not adequate for storing vast amount of configuration information.

- SoftwareAG Alfabet is also a repository that potentially could serve as a configuration management repository but again is not fit for purpose when it comes to more granular (and vast amount of) configuration items.
- Commercial CMDB implementations (e.g. BMC Atrium) are used to manage large landscapes and the level of details required for a CM implementation is again not fit for purpose. The granularity of CI items is often on the service/application level, and not on an individual product configuration property level.

At the bottom of the middle section of the diagram, is the required space to fill for the purpose of the CM Strategy and no commercial/open source utility has been identified. This is the space which OpenCM intends to fill.

3. Installation and Prerequisites

OpenCM comes as an Integration Server package and should preferably be deployed to an “administrational” type of server, separate/different from a business runtime IS.

3.1. Summary

The following steps are required to set up OpenCM and perform auditing:

1. Install OpenCM package on the Integration Server (this section)
2. Define the inventory (see section 5.1)
3. Define what to extract (see section 4.4)
4. Define what to audit (see section 4.5)

3.2. Compatibility

OpenCM has been developed and tested on webMethods version 9.9, 9.12, 10.0, 10.1, 10.3 and 10.5.

The ability to extract and compare target runtime versions is based on the respective SPM/SPM plugin capabilities. For example, certain information extracted from a v9.0 installation is not available compared to a 10.5 installation.

3.3. Download

OpenCM package is available to download from SoftwareAG GitHub Repository

3.4. Dependencies & Requirements

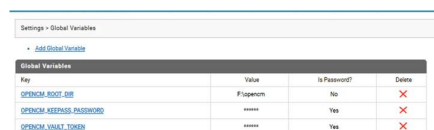
- ✓ OpenCM is reliant on webMethods Command Central Platform Managers (SPM's) to extract and update runtime configuration properties.
- ✓ OpenCM also utilizes file system storage for property configuration information: allocate a separate folder/directory on a file system that can be accessed by OpenCM. Estimate 3-4 Mb of disk space for each managed server in the landscape.

3.5. Installation

1. Install OpenCM on an Integration Server using the regular package installation procedures.
2. To avoid overwriting configuration files (and repository extracts) when a new OpenCM package is deployed, create a separate “root” directory on the file system for opencm specific configuration files and storage of extracted property files. E.g.:
 - OpenCM directory = F:\SoftwareAG\opencm

By default, OpenCM stores all configuration files within its package\resources directory. It makes sense to copy that folder to the external directory.

3. Update the global variable “OPENCM_ROOT_DIR” and reload the package



Key	Value	Is Password?	Delete
OPENCM_ROOT_DIR	F:\opencm	No	
OPENCM_EXTRACT_PASSWORD	*****	Yes	
OPENCM_VERIFY_TOKEN	*****	Yes	

4. Using OpenCM

The following section describes how to use OpenCM. To reach the OpenCM home page, open up a browser and access the Integration Server where the package was installed. Add /OpenCM to the URL.

4.1. Inventory

As described, the Inventory is a definition of all installations in the landscape. It is also a pre-requisite for being able to extract information from the runtime nodes (which then can be audited).

The landing page of OpenCM UI is just that, - the inventory.

The Inventory is a visualization of all the integration landscape installations. Select from left side inventory tree.

Notes:

- A single installation defined in the inventory, may or may not have been "inspected" (i.e. configuration properties extracted).
- If it has been extracted, then it will exist within the OpenCM configuration **repository**. The inventory installation table will then contain additionally decorated information, such as:
 - Version
 - Installation Date
 - Operating system
 - Number of CPUs
 - etc.
- An installation in **bold** indicates whether an installation is in the repository. Click on the installation name to view the repository.
- To include or exclude certain information in the inventory table, press the "Columns" button.
- Pressing the left-side "plus-sign" in the table list displays port information for the installation. Again, if there is no repository for the installation, only the ports defined in the inventory will be shown.

On the left side of the page is a tree-structure of installation groups (or categories). I.e. it is possible to group multiple installations together in a meaningful way. For example, all installations in the development environment can be found under the "Dev Zone" folder group.

The tree structure is completely flexible in the sense that any number of nested groups can be defined. The only requirement is that a group cannot contain both installations and groups at the same time (it doesn't make sense either).

Clicking on any group on the left side tree will then show all the installations defined underneath:

Ports	Inventory Location	Hostname	Installation	Version	Description
	Acme->Dev Zone->Central Admin	srv124.acme.local	ACME_DEVZ_ADM_V103_01		Central DEV Admin IS
	Acme->Dev Zone->Central Admin	srv123.acme.local	ACME_DEVZ_ADM_V103_CCE_01		
	Acme->Dev Zone->Database Servers	db311.acme.local	ORA_V120		Oracle version 12g for Development
	Acme->Dev Zone->Database Servers	db345.acme.local	MSSQL_V2016		MS SQL Server 2016
	Acme->Dev Zone->Database Servers	db369.acme.local	DB2_V123		DB2
	Acme->Dev Zone->Local Dev	srv124.acme.local	ACME_LDEV_S124_V104_01	10.4.0.0.189	Used by John Smith
	Acme->Dev Zone->Local Dev	srv125.acme.local	ACME_LDEV_S125_V104_01	10.4.0.0.189	
	Acme->Dev Zone->System Test->ESB->Batch Cluster	srv134.acme.local	ACME_SYS_ESB_V103_BATCH_01		
	Acme->Dev Zone->System Test->ESB->Batch Cluster	srv135.acme.local	ACME_SYS_ESB_V103_BATCH_02		
	Acme->Dev Zone->System Test->ESB->Online Cluster	srv137.acme.local	ACME_SYS_ESB_V103_ONLINE_01		

Showing 1 to 10 of 20 entries

External Connections: Product DBs | JDBC Connections | SAP Connections | Messaging

The resulting table is a list of all the installations that have been defined under the "Dev Zone" group.

The top information bar shows how many unique servers are included in the selection and in which domains they belong.

The table underneath shows the details of each installation, along with tree location (column 02), server name (column 03), installation node name (column 04), etc. There are multiple columns in the table with or without information, which is dependent on whether the installation in question has been extracted or not.

In other words, if an installation has been extracted, the inventory table has been decorated with additional information that is only known if the properties have been retrieved. For example, the version of the webMethods installation can be seen under the Version column and only a couple of installations have that value populated.

4.1.1. Searching the Inventory

On top of the inventory table on the lefthand side, is a search box. Enter any text and the table will automatically be filtered out based on the search criteria. The text entered will be matched with any of the displayed column values.

Any search on other OpenCM pages where tables are present is behaving similarly.

4.1.2. External Connections

Below the inventory table are additional buttons that in essence show information from the extracted properties in a more user-friendly format.

The screenshot displays the OpenCM webMethods Configuration Management interface. On the left is a tree view showing the hierarchy: OpenCM > Acme > Dev Zone > Local Dev > ACME_DEV_S124_V104_01. The main area contains two tables. The top table, titled 'Total Servers', shows 2 servers for 'acme.local' and 0 for 'Unqualified'. Below it is a search box and a table with columns: Ports, Inventory Location, Hostname, Installation, Version, and Description. The table shows two entries for 'Acme-Dev Zone-Local Dev' with hostnames 'srv124.acme.local' and 'srv125.acme.local', both with installation 'ACME_DEV_S124_V104_01' and version '10.4.0.0.109'. Below the table are buttons for 'External Connections': Product DBs, JDBC Connections, SAP Connections, and Messaging. The bottom table, titled 'Product DBs', shows two entries for 'ACME_DEV_S124_V104_01' and 'ACME_DEV_S125_V104_01', both with ID 'Embedded_Database_Pool' and URL 'jdbc:derby:embedded:create=true'. The table also has columns for Username and Username.

Ports	Inventory Location	Hostname	Installation	Version	Description
	Acme-Dev Zone-Local Dev	srv124.acme.local	ACME_DEV_S124_V104_01	10.4.0.0.109	Used by John Smith
	Acme-Dev Zone-Local Dev	srv125.acme.local	ACME_DEV_S125_V104_01	10.4.0.0.109	

Installation	ID	URL	Username
ACME_DEV_S124_V104_01	Embedded_Database_Pool	jdbc:derby:embedded:create=true	APP
ACME_DEV_S125_V104_01	Embedded_Database_Pool	jdbc:derby:embedded:create=true	APP

By selecting the “Product DBs”, a list of defined database connections are displayed in the lower table. This will help identifying which databases are used by the selected installation nodes.

Similarly, other external connections are available:

- JDBC Connections
- SAP Connections
- Messaging connections

4.2. Repository

As mentioned above, if an installation has been extracted, i.e. when properties from the remote installation have been collected, it is possible to get a more detailed view into those, - in the so-called configuration “repository”.

The repository is viewed by clicking on the node name from within the inventory table.

Installation Repository

Return to Inventory

ACME_LDEV_S124_V104_01

- integrationServer-default
- OSGI-IS-default-EventRouting
- OSGI-SPM-EDA-DEPLOYER
- OSGI-SPM-ENGINE
- OSGI-IS-default
- OSGI-SPM
- OSGI-IS-default-DigitalEventServices
- NODE-PRODUCTS
- NODE-FIXES

Select Component from tree

The Repository contains all extracted configuration properties for an installation.

Clarifications:

- The repository for an installation is made up of:
 - Components
 - Instances
 - Properties
- The components are listed in the left-side tree. Clicking a component will show the instances in a table.
- The instances are then also clickable, which will display all the properties for a single instance.
- An installation itself also contains certain properties (which are displayed here).

Installation Properties

Search:

Property Key	Value
code	V104
cpuArchitecture	amd64
cpuCores	12
cpuType	PCD
displayName	Windows 10
hardwareId[0]	3C-F8-11-22-00-79
hardwareId[1]	50-40-40-4F-4E-43
lastCollectingTimeStamp	

Showing 1 to 8 of 20 entries

The repository view shows all the properties for a single installation and are structure as a tree model:

- Installation
 - Component
 - Instance
 - ⇒ Property Key + Value

The notation is identical to how the Software AG Platform Manager (SPM) structures its content, although the OpenCM provides a somewhat different view into the configuration properties.

A single installation can thus have multiple components, as listed on the left side of the user interface. By clicking on a single component, the available instances for that component is displayed on the top center table.

Installation Repository

Return to Inventory

ACME_LDEV_S124_V104_01

- integrationServer-default
- OSGI-IS-default-EventRouting
- OSGI-SPM-EDA-DEPLOYER
- OSGI-SPM-ENGINE
- OSGI-IS-default
- OSGI-SPM
- OSGI-IS-default-DigitalEventServices
- NODE-PRODUCTS
- NODE-FIXES

Instances for component: integrationServer-default

Search:

Instance Name
COMMON-ADMINUI
COMMON-CLUSTER
COMMON-COMPONENT-ENDPOINTS-local
COMMON-COMPONENT-ENDPOINTS-testServer
COMMON-CONFIGURATION-TYPE-METADATA-com.softwareag.platform.management.is.configuration.impl.ISResourceMetadata.xml
COMMON-CONFIGURATION-TYPE-METADATA-com.softwareag.platform.management.is.configuration.impl.WSConsumerEndpointsMetadataComponent.xml
COMMON-CONFIGURATION-TYPE-METADATA-com.softwareag.platform.management.is.configuration.impl.WSProviderEndpointsMetadataComponent.xml
COMMON-CONFIGURATION-TYPE-METADATA-com.softwareag.pim.spm.spm.impl.CommonComponentEndpointsConfigurationTypeMetadata.xml

Showing 1 to 8 of 124 entries

The example above shows all the instances for the component “integrationServer-default”, which is the Integration Server runtime. By clicking on a single instance, all the properties available for that instance are displayed in the table on the lower table beneath.

Selecting the “IS-RESOURCES” instance shows the following properties:

The screenshot displays the OpenCM webMethods Configuration Management interface. At the top, there is a navigation bar with 'Return to Inventory' and 'Installation Repository'. Below this, a sidebar on the left lists various components and instances, including 'ACME_LDEV_S124_V104_01', 'integrationServer-default', and 'OSGI-SPI-ENGINE'. The main area shows a search bar and a table of instances for the component 'integrationServer-default'. The table lists instances: 'IS-PRIMARYPORT', 'IS-QUIESCEPORT', 'IS-RESOURCES', and 'IS-SVSPROPS'. Below the table, there is a section titled 'Properties for instance: IS-RESOURCES' which contains a table with two columns: 'Property Key' and 'Value'. The table lists properties such as 'attributes', 'configurationTypeid', 'description', 'displayName', 'id', 'Resources.ServerThreadPool.AvailableThreadsWarningThreshold', 'Resources.ServerThreadPool.MaximumThreads', and 'Resources.ServerThreadPool.MinimumThreads'. The 'Value' column shows the corresponding values for these properties.

Property Key	Value
attributes	
configurationTypeid	IS-RESOURCES
description	Integration Server Resource Settings
displayName	Resource Settings
id	IS-RESOURCES
Resources.ServerThreadPool.AvailableThreadsWarningThreshold	15
Resources.ServerThreadPool.MaximumThreads	75
Resources.ServerThreadPool.MinimumThreads	10

The lower properties table contain two columns:

- Property Key
- Property Value

Subsequent audits are defined by specifying the component names, the instance names and the property keys so it is important to understand the repository structure above.

4.3. Anonymous Access

Stakeholders that are interested in viewing the inventory, and associated repository content, can gain access by configuring some services as “Anonymous”. Other Administration commands should be available only after successful login to the hosting Integration Server.

To provide anonymous access, the following services Execute ACL should be updated:

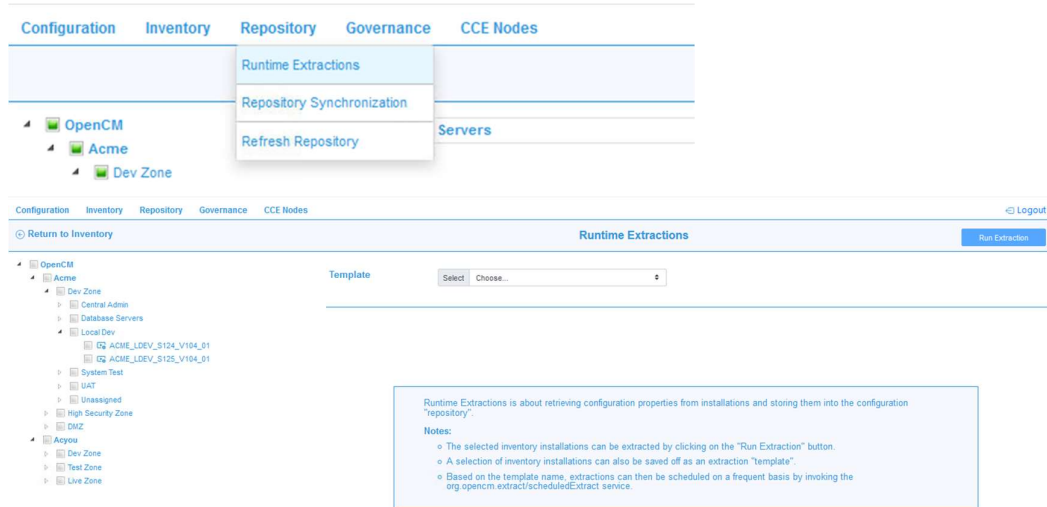
- org.opencm.pub.inventory.getInventory
- org.opencm.pub.repository.getRepository

4.4. Extractions

In order to perform extractions, one needs to log on to OpenCM to access all administrative functions. This is done by clicking on the Administration link on the top right corner of the UI:

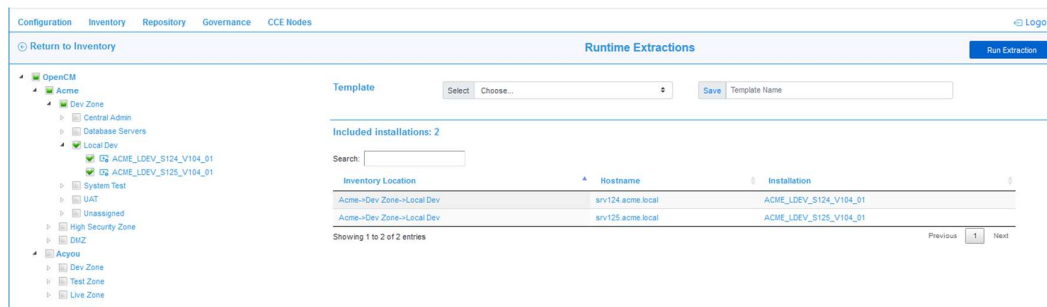


Once logged on, a menu is now available, - select “Runtime Extractions”:



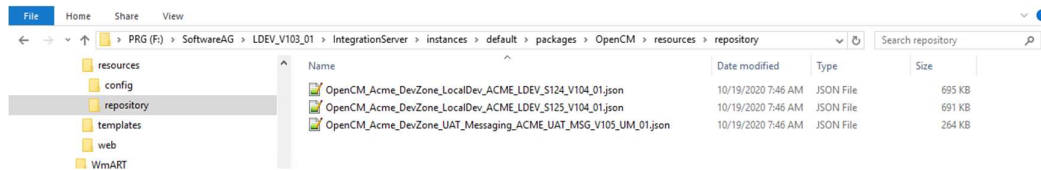
The tree layout on the left hand side reflect the installation nodes from the inventory, similarly as on the main landing page where the inventory is displayed.

By selecting groups in the tree, or individual installations, a table of selected installations is then displayed:



In this example, we have selected two installations to be extracted. To start, click on the “Run Extraction” button on the top left corner.

This will kick off an extraction job and is run in the background. The job itself will, based on the selected installations, connect to the remote installations and retrieve all the configuration properties it can find. When finished, it will store the content in the form of a json structure in the configured repository location on file system (in this case it is within the OpenCM package itself, but we recommend to use a separate, external location).



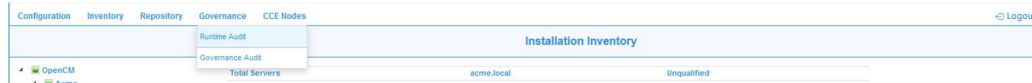
The file name of the extracted properties is based on the inventory path to where the installation resides. This will ensure that all installations are uniquely identified, even though there might be multiple installations with the same name.

4.5. Auditing

Auditing comes in two forms:

- Runtime Audits
- Governance Audits

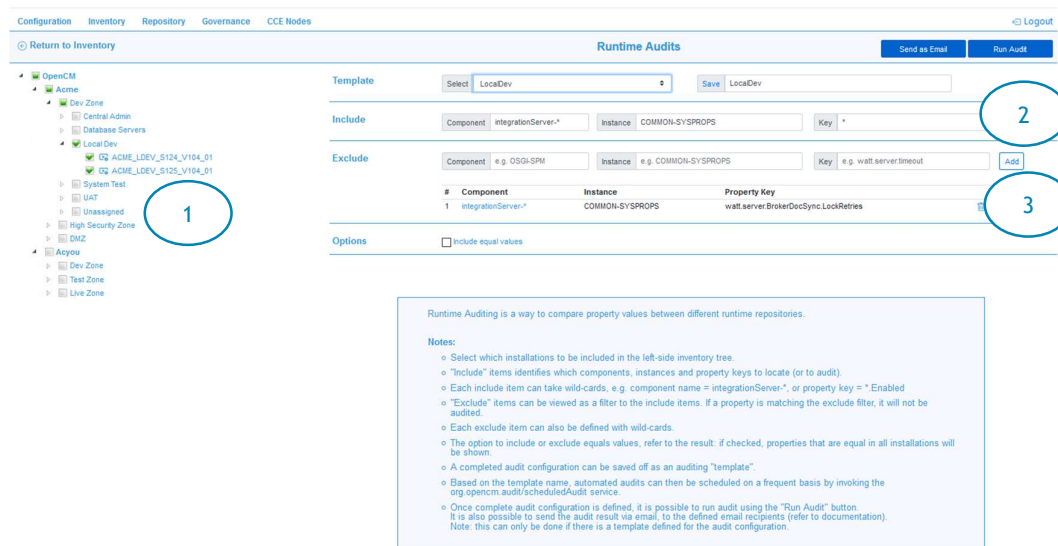
Both functions are reached via the “Governance” menu:



4.5.1. Runtime Audits

This function compares property definitions between multiple installations. The process for defining a runtime audit is as follows:

1. Select the desired installation nodes from the left side inventory tree
2. Define the repository properties that should be audited:
 - a. Component name
 - b. Instance name
 - c. Property key
3. Optionally, it is possible to exclude certain properties that would fall within above selection, sort of as a filter to the included properties. One can define multiple filters, and each filter follows the same notation as the include section, i.e. one would specify the component, instance and the property key to filter out.



In the above example, we have selected two installations, and we want to audit all the “extended settings” of these Integration Servers.

- The component name is defined as “integrationServer-*”, using a wildcard character to indicate that any integration server runtime component should be included (there might be non-default IS instance names).
- The instance name is defined as “COMMON-SYSPROPS”, which is the name of the instance for extended properties.

- For property key, we have indicated “*”, which means that we want to audit all property keys for this instance.

To perform the audit, click on the “Run Audit” button on the top right corner. The result of the audit will be shown in a table as follows:

#	Component	Instance	Property Key
1	integrationServer-*	COMMON-SYSPROPS	watt.serverBrokerDocSync.LockRetries

Options ☐ Include equal values

Runtime Audit Result:
Total number of properties audited: 572 - Total Differences: 2

Search:

Property	Location	Value
watt.net.timeout	ACME_LDEV_S124_V104_01->integrationServer-default->COMMON-SYSPROPS ACME_LDEV_S125_V104_01->integrationServer-default->COMMON-SYSPROPS	300 301
watt.security.cert.wmChainVerifier.trustByDefault	ACME_LDEV_S124_V104_01->integrationServer-default->COMMON-SYSPROPS ACME_LDEV_S125_V104_01->integrationServer-default->COMMON-SYSPROPS	true false

Showing 1 to 2 of 2 entries

Previous **1** Next

The table includes information about all the properties that are different, comparing the same property amongst all selected installation nodes. In this example, there were 572 properties audited, and 2 were different.

The location column indicates where the property key/value was found.

The value column shows the inspected value for the above location. In this case, we can see that the “watt.net.timeout” value was different between the two installations.

Other functions:

- **Options:** before running the audit, it is possible to include the complete result, not only the properties that were different. This can be useful when one wants to quickly inspect the property value for multiple installations at once.
- **Send as Email:** instead of showing the result in the user interface as above, the audit result will be sent via email to defined recipients (refer to Email configuration).
- **Template:** a defined audit configuration can be saved off for later use as a template. The list of templates are found in the drop-down box on top of the audit definitions. It is also possible to update an existing template by simply saving it off with the same name.

Note: to schedule automatic runtime audits, a template is required. The template name is the input to the job that runs the audit.

4.5.2. Governance Audits

This function is very similar to runtime audits, with the main difference being the property value:

- Each installation node that is selected, is compared with a “locked-down” property value that can be seen as the source of truth value.

I.e., certain properties are required to be of a certain value, regardless of what is defined and configured on the runtime installations. The configuration looks like this:

Configuration Inventory Repository Governance CCE Nodes

Return to Inventory Governance Audits Send as Email Run Audit Logout

Saved Rules: LocalDev01

Governance Rule

Component: e.g. OSQ-SPM Key: e.g. watt.server.timeout

Instance: e.g. COMMON-SYSPROPS Value: e.g. 60

#	Component	Instance	Key	Value
1	integrationServer-default	COMMON-SYSPROPS	watt.net.timeout	300
2	integrationServer-default	COMMON-SYSPROPS	watt.net.ftpClientDataConnTimeout	30000

Save Rule LocalDev01 (Selected installation nodes: 2)

To define a governance audit (rule), the following needs to be performed:

1. Select the desired installation nodes from the left side inventory tree
2. Define the repository properties that should be audited:
 - a. Component name
 - b. Instance name
 - c. Property key
 - d. Property value

It is possible to define multiple property key/values for a single rule.

The perform the audit by clicking on the “Run Audit” button and the result is displayed as table underneath:

1	integrationServer-default	COMMON-SYSPROPS	watt.net.timeout	300	
2	integrationServer-default	COMMON-SYSPROPS	watt.net.ftpClientDataConnTimeout	30000	

Save Rule LocalDev01 (Selected installation nodes: 2)

Governance Audit Result:
Total number of properties audited: 2 - Total Out of Compliance: 1

Search:

Property	Location	Value
watt.net.ftpClientDataConnTimeout	ACME_LDEV_S124_V104_01->integrationServer-default->COMMON-SYSPROPS ACME_LDEV_S125_V104_01->integrationServer-default->COMMON-SYSPROPS	30000 30000
watt.net.timeout	ACME_LDEV_S124_V104_01->integrationServer-default->COMMON-SYSPROPS ACME_LDEV_S125_V104_01->integrationServer-default->COMMON-SYSPROPS	300 301

Showing 1 to 2 of 2 entries Previous 1 Next

The result is similar to the runtime audits, with the main difference being the color-coded values displayed on the right-side column. All property values that conforms to the rule are colored green, whilst the values that are out of compliance are colored in red.

Other functions:

- **Send as Email:** instead of showing the result in the user interface as above, the audit result will be sent via email to defined recipients (refer to Email configuration).
- **Saving Rules:** each rule definition (i.e. which installations and what properties to audit) can be saved off with a rule name.

Note: to schedule automatic runtime audits, a template is required. The template name is the input to the job that runs the audit.
- **Running multiple audits at once:** run the governance audit by selecting multiple defined rules in the “Saved Rules” box.

4.6. Command Central Node Definitions

When setting up or maintaining Command Central, each installation that the particular CCE will manage must be defined. Each definition must be provided with the following:

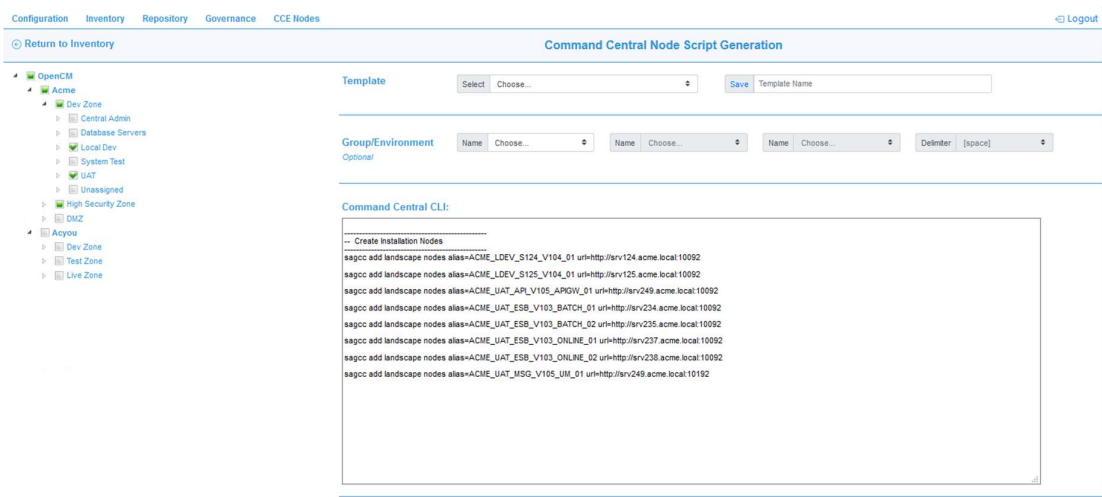
- A unique installation node name
- The server name of where the installation is located
- The port number which SPM is running on
- The username and password of the remote SPM that CCE connects to

In addition, when having multiple installation nodes within CCE, it makes sense to group together so they can more easily be located. This is done via so-called “environments”, or groups.

OpenCM already has all the information required in its inventory, so it makes sense to automate (or provide a script) that can generate the node definition for Command Central. This is done via the “CCE Nodes” menu item:



To create the CLI used for Command Central, simply select all the nodes that should be used:



The Command Central CLI text box area will then be filled with the CLI commands necessary to define them.

Optionally, groups can also be created by defining a group name. The group is constructed by the location within the inventory tree, and selected installation nodes will then be part of that group.

It is possible to define up to three separate levels:

The screenshot displays the 'Command Central Node Script Generation' tool within the OpenCM webMethods Configuration Management interface. The left sidebar shows a tree view of the configuration hierarchy, including 'OpenCM', 'Acme', 'Dev Zone', 'Central Admin', 'Database Servers', 'Local Dev', 'System Test', 'UAT', 'Unassigned', 'High Security Zone', 'DMZ', 'Acyou', 'Dev Zone', 'Test Zone', and 'Live Zone'. The main workspace is titled 'Command Central Node Script Generation' and features a 'Template' section with a 'Select' dropdown and a 'Save' button. Below this is a 'Group/Environment' section with three columns for 'Name', 'Tree Level 01', 'Tree Level 02', and 'Tree Level 03', and a 'Delimiter' dropdown menu. The 'Delimiter' menu is open, showing options like '[space]', ';;', '->', '>>', '!', '-', '#', and '@'. The 'Command Central CLI' section contains a script for creating environment groups and installation nodes, using the 'Acme ==> Dev Zone ==> LocalDev' group structure.

As shown in the example, we have defined level 1, level 2 and level 3 to be part of the group name. All the groups will then be created in Command Central first, and then each associated installation node will be added to that group.

The **delimiter** drop-down box indicates a sequence of characters to use to separate the different levels. In this example we have opted for “=>” and the group name as shown in Command Central will then be something like:

- Acme ==> Dev Zone ==> LocalDev

All installations that fall underneath above group (in the inventory) will then be associated with this Command Central group.

5. Managing OpenCM

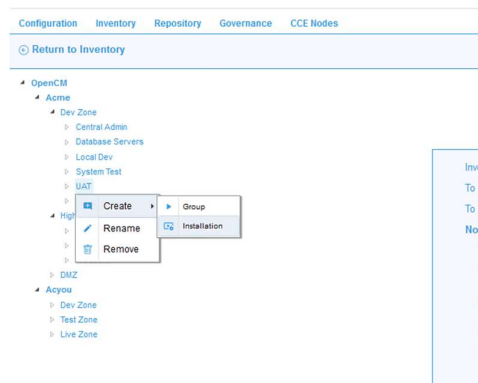
The above section describes how to make use of OpenCM, but there are other functions to support configuring and administrating OpenCM.

5.1. Managing the Inventory

The inventory is managed from the following menu item:



From within “Manage Inventory”, it is possible to define groups in the tree structure and define individual installation nodes.



In general, there is no limitation on how the tree is defined except for the following:

- It is not possible to combine both installation nodes and group on the same level.
- It is not possible to define multiple installation nodes with the same name within the same group

I.e., a group cannot contain both installation nodes and at the same time have sub-groups underneath. Also, it is not possible to have an installation with the same name as another installation node within the same group. It is possible to have the same name for installations, but then they must be defined under different groups.

- o **It is recommended to use unique names for all installations however.**

For example, defining two separate installations with the same name within Command Central will not work, since it requires unique names.

5.1.1. Defining an Installation

An installation comes with a number of configuration settings that must be defined (to make it useful). For example, what is the server name it is hosted on.

By selecting an installation node in the tree, the rightside portion of the page displays all the settings that can be entered:

- **Server hostname:** makes sense to enter in the fully qualified name
- **Description:** any arbitrary text that can be useful later (searchable from inventory page)
- **One or multiple runtimes:** when connecting to a webMethods installation, the SPM endpoint is necessary to define. The name of the component is identical to the component name as used by the SPM name itself (i.e. “OSGI-SPM”). Only one SPM per installation is possible. In addition, it is also possible to add other runtimes, such as an Integration Server runtime, e.g.:
 - integrationServer-default
 - integrationServer-myInstance

The Integration Server runtimes are used to connect and collect information that is currently not possible to extract using the SPM. For example, to collect information about installed packages and their associated versions, a direct connection to the target Integration Server is established via the IS Admin UI.

- **Username and password:** these are the credentials needed to make a remote connection to the target runtime. (Please also refer to the Secrets configuration section in this document). Important to know is that no passwords are stored in clear text within the OpenCM configuration store.
- Optionally, it is possible to add load balancer information to the installation node (if such exists). This is just to be able to provide additional port information when viewing the inventory.

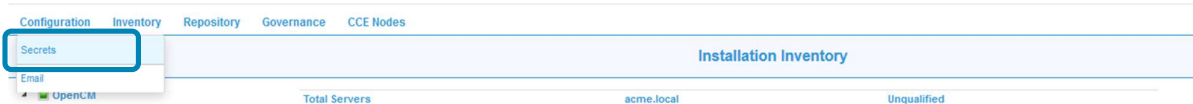
Note: it is possible to navigate around the inventory tree, define new groups, new installations or change existing ones without having to save each time. All inventory information is stored locally until the inventory definitions are saved off (using the “Save Inventory” button on the top right corner).

5.2. Secrets

Passwords that are needed by OpenCM to connect to remote installations need to be defined and stored somewhere. OpenCM supports two different mechanisms:

1. Using a local Keeppass file (default)
2. Using a central Vault (HashiCorp)

Secrets configuration can be found here:



To use the local Keepass mechanism, enter in the following:

Secrets Configuration defines where to store/retrieve passwords from.
By default, OpenCM uses a local Keepass database for storing secrets

Notes:

- Local Keepass means that passwords are entered when defining the inventory (runtimes). I.e. the entered password will be written to the secrets local store into the path derived from the inventory path.
 - When Local Keepass is used, the password to the keepass db is using the IS Global Variables key "OPENCM_KEEPPASS_PASSWORD"
- Vault involves integration with a central Vault (Hashicorp) system where passwords must have already been created
 - Vault URL is the remote location of the Vault.
 - When Vault is used, the Token used to gain access is using the IS Global Variables key "OPENCM_VAULT_TOKEN"
 - When defining Inventory (runtimes), the Vault path to the secret is specified.

The Master Password refers to the Keepass database and must be the same. The definition here is only used to open the database file, and changing the password must therefore be accompanied by changing the master password of the Keepass file itself. By default, the password is "manage" and should be changed.

- The location of the Keepass file is config/secrets/secrets.kdbx

The other mechanism is Vault, which is configured accordingly:

- **Token** is the secret token that allows access to the central Vault system
- **URL** is the endpoint to the Vault
- **Version** is the version of the used secrets engine inside Vault.

If Vault integration is used, then there will be no passwords stored locally within OpenCM. Instead they are identified by password handles, and reflect the Vault path where the secret is stored. This path will then show up instead of the password field when managing the inventory.

5.3. Email Configuration

Configuring email allows for auditing results to be sent as opposed to shown directly in the user interface. It is also a pre-requisite for using scheduled audits where the result is put into an email after execution.

Email configuration can be found here:

Configuration	Inventory	Repository	Governance	CCE Nodes
Secrets	Installation Inventory			
Email				
Acme				
Total Servers		acme.local		Unqualified
2		2		0

Within the Email Configuration page, define the appropriate fields:

Configuration Inventory Repository Governance CCE Nodes

Return to Inventory Email Configuration Save Configuration

Logout

Email Configuration is used for sending audit result emails to defined recipients.

Notes:

- Email configuration is used in the following situations:
 - Scheduled Audits - results will be sent to recipient list
 - Audits from UI - results can be sent to recipient list

SMTP Server

Hostname: smtp.gmail.com Port: 587 Start TLS: true

Access

Username: mata.hakim.hansson@gmail.com Password: *****

Mail

From Email: audit@opencm.org Subject: OpenCM Audit

Properties

#	Key	Value
1	mail.smtp.auth	true

Recipients

#	Email
1	hakim.hansson@softwareag.com

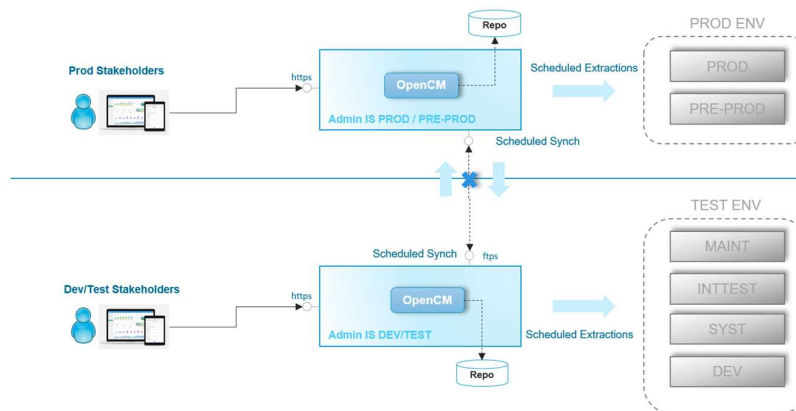
The recipients list indicate the email addresses of the person/people that will receive auditing results. The example above shows testing configuration against a gmail smtp server.

5.4.Repository Synchronization

In situations where there are multiple OpenCM installations, each responsible for collecting information from certain installations (for example in different network domains), it is possible to synchronize the extracted configuration properties with one another.

This is however not the recommended approach for keeping multiple OpenCM installations in synch: use instead a separate code control repository, such as git, to push and pull configurations and repository content. This way it is possible to also include all sorts of configurations done on one OpenCM instance and make it available on another.

If that is not an option, there is a way to synchronize repositories via FTPS, and would thus require firewall opening in between the two.



As shown, the target OpenCM (Integration Server) must be configured with an FTPS port and made accessible from the sending node.

In addition, the target username/password, server and port must be made available for the sender. This is done via the inventory management page and by including a runtime named OPENCM-SYNCH:

Runtimes

Component: Protocol: Port:

Alt Hostname: Username: Password:

#	Name	Protocol	Port	Alt Hostname	Username	Password	
1	OSGI-SPM	http	10092		Administrator	*****	<input type="button" value="Delete"/>
2	OPENCN-SYNCH	ftps	10021		Administrator	*****	<input type="button" value="Delete"/>

All targets that include such a runtime will be listed in the below target drop-down list:

Repository Synchronization

Template:

Target:

ACME_DEVZ_ADM_V103_JS_01
ACME_SECZ_ADM_V103_JS_01

Repository Synchronization involves sending extracted configurations from this OpenCM instance to another one.

Notes:

- » The "Target" represents the remote OpenCM instance to send to.
- » Targets are identified by inventory installations that contain the "OPENCN-SYNCH" entry.
- » Transmission protocol is **ftps** and the target must be reachable on the defined ftps port.
- » Which installations to synchronize is based on the selected inventory installations on the left-side tree.
- » To start synchronization, click on the "Run Synch" button. Only installations with repositories will be synchronized.
- » A selection of inventory installations (and selected target) can also be saved off as an synchronization "template".
- » Based on the template name, synchronizations can then be scheduled on a frequent basis by invoking the `org.opencm.synch.scheduledSynch` service.

Including a target, as well as selecting which nodes should be synchronized from the left-hand tree, the synchronization process can start (by clicking on the "Run Synch button").