DevOps for AEM as a Cloud Service

Environment Run Modes

Adobe

Student Workbook

Find your journey at learning.adobe.com >

Contents

DevOps for AEM as a Cloud Service

Adobe, 345 Park Avenue, San Jose, California 95110, USA.

10-14-2020

Introduction

You often need to define specific Adobe Experience Manager (AEM) behaviors for different types of AEM instances. For example, you need to define different behaviors for author instances that run in a development environment as compared to publish instances that run in a production environment . The Run modes in AEM enables you to define different behaviors.
The ACS AEM Commons is a suite of AEM components , services, and other tools for bootstrapping an AEM implementation.

Objectives

After completing this module, you will be able to:

- Explain supported Run Modes for AEMaaCS
- Start AEM with a desired Run Mode
- Create a configuration for the publish Run Mode
- Describe ACS Commons
- Deploy the ACS Commons package to a specific AEM environment type

Run Modes: An Overview

A basic set of configuration parameters is applied to all Run Modes. You can also define additional configuration parameter sets to meet your specific environment. These settings are applied as required.

Run Modes are typically defined to include the tier or service (author and publish) and the environment (dev, stage, prod).

The tier Run Modes are used at the installation time and then fixed for the entire lifetime of the service. They cannot be changed. The author and publish tier Run Modes are provided out- of- the- box.

The author and publish Run Modes are mutually exclusive. Any instance can be an author or publish instance, but not both. You can combine the installation Run Modes with other environment-specific run modes.

All configuration definitions are stored in the repository and activated by setting the appropriate Run Mode.

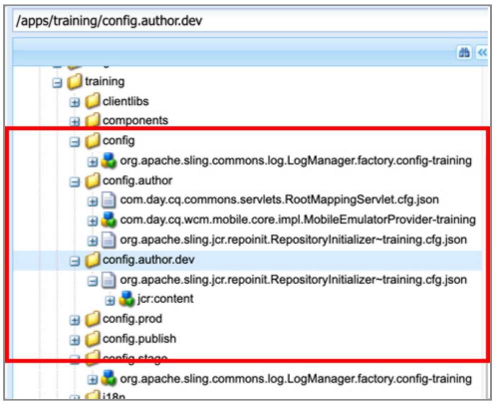
OSGi Configuration Run Modes

OSGI configuration Run Modes reference dev, stage, and prod for the environment or author, publish for the service. The supported formats for OSGi configuration Run Modes are:

config.<tier>

config.<tier>.<environment>

The service and environment must be specified in the supported order. For example, the order should be config.author.dev or config.publish.prod, as shown:

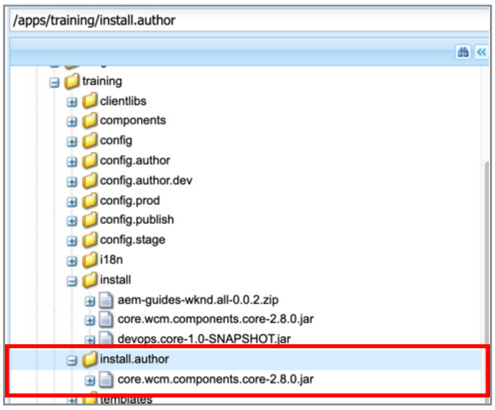


```
/apps/training/config.author.dev

training
  clientlibs
  components
  config
    org.apache.sling.commons.log.LogManager.factory.config-training
  config.author
    com.day.cq.commons.servlets.RootMappingServlet.cfg.json
    com.day.cq.wcm.mobile.core.impl.MobileEmulatorProvider-training
    org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json
  config.author.dev
    org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json
      jcr:content
  config.prod
  config.publish
  config.stage
    org.apache.sling.commons.log.LogManager.factory.config-training
  i18n
```

OSGi Bundle and Content Package Run Modes

OSGI bundle Run Modes are limited to the tier (author, publish). Per Run Mode OSGI bundles should be installed in a content package either under **install.author** or **install.publish** folders. For example, under **/apps/training/install.author/core.wcm.components.core-2.8.0.jar**, as shown:



/apps/training/install.author

Supported Run Mode Configurations

The supported Run Mode configurations in AEM as a Cloud Service(AEMaaCS) are:

- **config** (The default, applies to all AEM Services)
- **config.author** (Applies to all AEM Author service)
- **config.author.dev** (Applies to AEM Dev Author service)
- **config.author.stage** (Applies to AEM Staging Author service)
- **config.author.prod** (Applies to AEM Production Author service)
- **config.publish** (Applies to AEM Publish service)
- **config.publish.dev** (Applies to AEM Dev Publish service)
- **config.publish.stage** (Applies to AEM Staging Publish service)
- **config.publish.prod** (Applies to AEM Production Publish service)
- **config.dev** (Applies to AEM Dev services)
- **config.stage** (Applies to AEM Staging services)
- **config.prod** (Applies to AEM Production services)

Exercise 1: Run an AEM service with the 'dev' Run Mode

Scenario: To test the code behavior in an AEM as a Cloud Service developer environment, a developer wants to start a local AEM author instance with the 'dev' run mode.

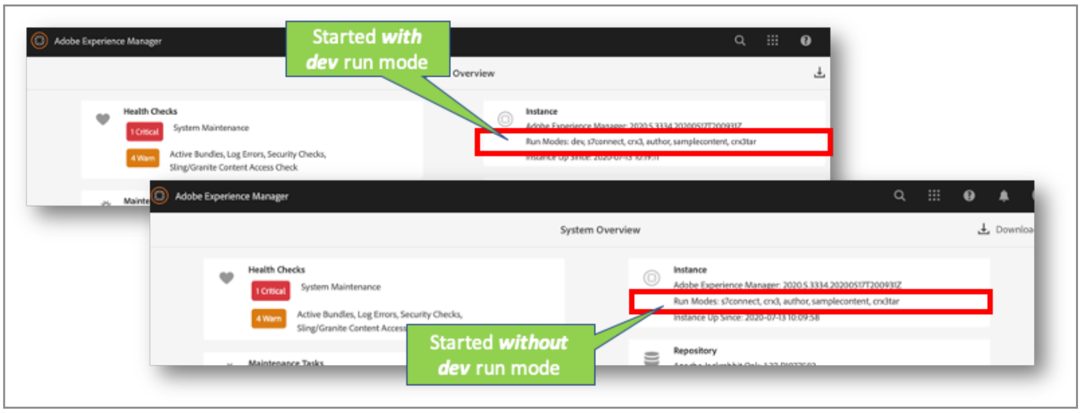
Prerequisites:

- AEM SDK
- Local AEM author service

You can validate whether your local AEM author instance has been started with the. 'dev' Run Mode. To do so:

1. From the Navigation page, navigate to **Tools** > **Operations** > **System Overview**.

2. Examine the list of run modes in the **Instance** block, as shown:



Started *with* *dev* run mode

Instance
Adobe Experience Manager: 2020.5.3334.20200517T200931Z
Run Modes: dev, s7connect, crx3, author, samplecontent, crx3tar

Started *without* *dev* run mode

Instance
Adobe Experience Manager: 2020.5.3334.20200517T200931Z
Run Modes: s7connect, crx3, author, samplecontent, crx3tar
Instance Up Since: 2020-07-13 10:09:58

If 'dev' is in the list of Run Modes, go to Exercise 2. If 'dev' is not in the list of Run Modes, follow the remaining steps in this exercise.

3. Shutdown AEM by clicking the **on/off** button on the gui window.

4. Open a command line window in the AEM author folder and enter the following command to start the AEM author instance with the 'dev' Run Mode:

```
$ java -jar aem-sdk-quickstart.jar -r author,dev -gui
```

Note: The **-gui** parameter opens the AEM gui window.

Exercise 2: Create Runmode-specific configs

Scenario: A developer wants to define the root mapping for author instances running in the development environment to be the Projects root folder.

Prerequisites:

- AEM SDK
- Local AEM author service running with dev run mode
- AEM project created from Maven AEM Project Archetype
- AEM project imported into Eclipse
- Maven 'Run As" Profiles defined

This exercise includes the following tasks:

1. Examine runmode-specific config folders created by Maven AEM project Archetype

2. Verify the configured value of Day CQ Root Mapping

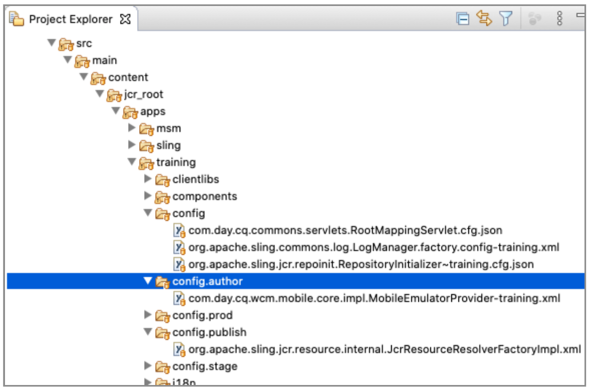
3. Create Run Mode-specific configurations

4. Deploy and validate the configuration

Task 1: Examine runmode-specific config folders created by Maven AEM project Archetype

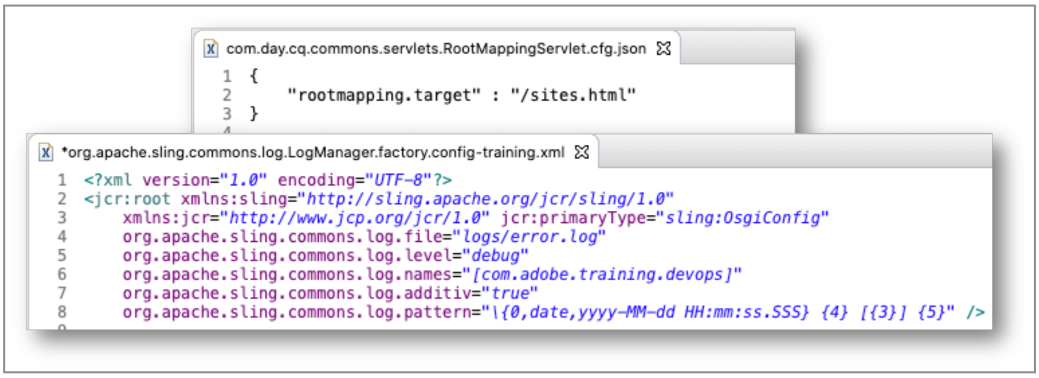
1. Using **Eclipse**, in your AEM Project, navigate to **ui.apps** > **src** > **main** > **content** > **jcr:root** > **apps** > **training**.

2. You will notice several **config.*** folders that hold OSGi configurations. For example: **config, config.author, config.publish, config.prod, config.stage** folders, as shown:



```
src
  main
    content
      jcr_root
        apps
          msm
          sling
          training
            clientlibs
            components
            config
              com.day.cq.commons.servlets.RootMappingServlet.cfg.json
              org.apache.sling.commons.log.LogManager.factory.config-training.xml
              org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json
            config.author
              com.day.cq.wcm.mobile.core.impl.MobileEmulatorProvider-training.xml
            config.prod
            config.publish
              org.apache.sling.jcr.resource.internal.JcrResourceResolverFactoryImpl.xml
            config.stage
            i18n
```

3. Open a few of the folders and investigate the OSGi configuration definitions, as shown:



com.day.cq.commons.servlets.RootMappingServlet.cfg.json

```
{
    "rootmapping.target" : "/sites.html"
}
```

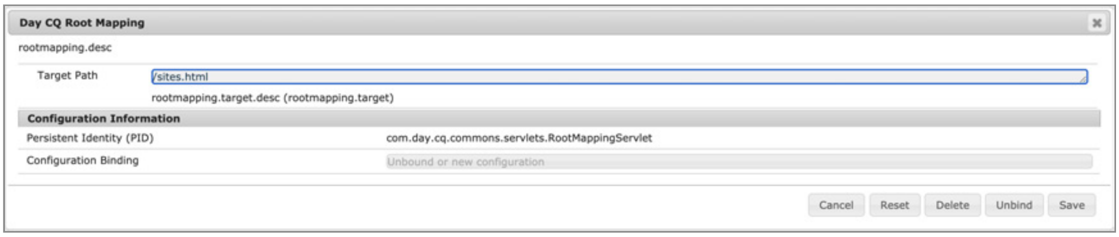
*org.apache.sling.commons.log.LogManager.factory.config-training.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<jcr:root xmlns:sling="http://sling.apache.org/jcr/sling/1.0"
    xmlns:jcr="http://www.jcp.org/jcr/1.0" jcr:primaryType="sling:OsgiConfig"
    org.apache.sling.commons.log.file="logs/error.log"
    org.apache.sling.commons.log.level="debug"
    org.apache.sling.commons.log.names="[com.adobe.training.devops]"
    org.apache.sling.commons.log.additiv="true"
    org.apache.sling.commons.log.pattern="\{0,date,yyyy-MM-dd HH:mm:ss.SSS} {4} [{3}] {5}" />
```

Task 2: Verify the configured value of Day CQ Root Mapping

1. In a browser window, from the Navigation page, navigate to **Tools** > **Operations** > **Web Console** to open the Web Console Config Manager.

2. Search on the web page for **Day CQ Root Mapping**, as shown:



Day CQ Root Mapping

rootmapping.desc

Target Path: /sites.html

rootmapping.target.desc (rootmapping.target)

Configuration Information

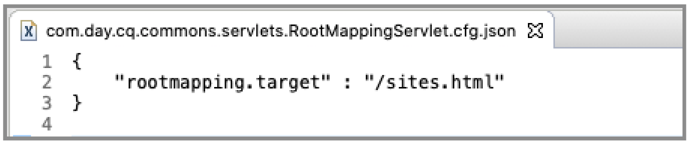
Persistent Identity (PID): com.day.cq.commons.servlets.RootMappingServlet

Configuration Binding: Unbound or new configuration

Notice that the Target Path is /sites.html.   Day CQ Root Mapping was configured to have this value.

3. In **Eclipse**, navigate to **ui.apps** > **src** > **main** > **content** > **jcr:root** > **apps** > **training** > **config**.

4. Open **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** and inspect the contents, as shown:



com.day.cq.commons.servlets.RootMappingServlet.cfg.json

```
{
    "rootmapping.target" : "/sites.html"
}
```

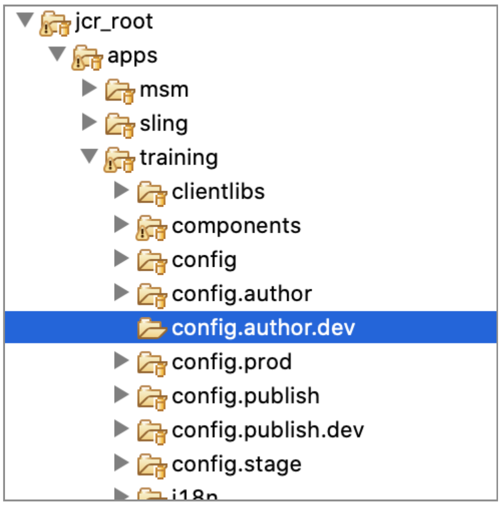
The **Day CQ Root Mapping** has taken this value because the configurations in the /apps/training/config folder apply to all AEM instances, regardless of the run modes.

Task 3: Create Run Mode-specific configurations

1. In Eclipse, navigate to **ui.apps** > **src** > **main** > **content** > **jcr:root** > **apps**.

2. Right-click the **training** node and select **New** > **Folder** from the context menu.

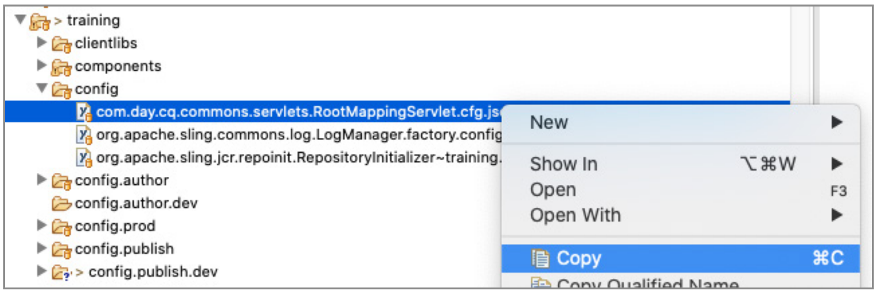
3. Enter **config.author.dev** for the folder name and click **Finish**, as shown:



Next, create a **Day CQ Root Mapping** configuration for the dev Run Mode.

4. Navigate to **ui.apps** > **src** > **main** > **content** > **jcr:root** > **apps** > **training** > **config**.

5. Right-click the **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** file and click **Copy**, as shown:


> training
clientlibs
components
config
com.day.cq.commons.servlets.RootMappingServlet.cfg.js
org.apache.sling.commons.log.LogManager.factory.config
org.apache.sling.jcr.repoinit.RepositoryInitializer~training.
config.author
config.author.dev
config.prod
config.publish
> config.publish.dev
New
Show In ⌥⌘W
Open F3
Open With
Copy ⌘C
Copy Qualified Name

6. Right-click the **config.author.dev** folder and click **Paste**.

7. Open the **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** file that you just pasted in the **config.author.dev** folder.

8. Modify the contents to point the **rootmapping.target** property to the English language root of the **US** site in **WKND**, as shown:

```
{
    "rootmapping.target" : "/projects.html"
}
```

Task 4: Deploy and validate the configuration

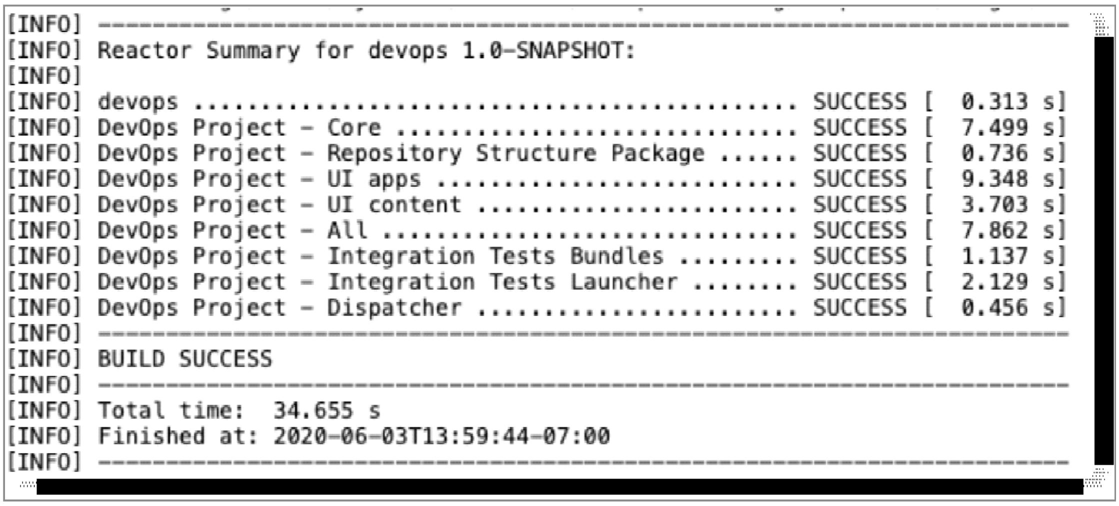
You first need to deploy to the author server. To do so:

1. If you do not already have a command line window open in your < AEM Project > directory, open one now.

2. Enter the following command to deploy your changes to the local AEM author instance:

```
$ mvn clean install -P adobe-public -P autoInstallSinglePackage
```

3. Verify that the build completed successfully, as shown:



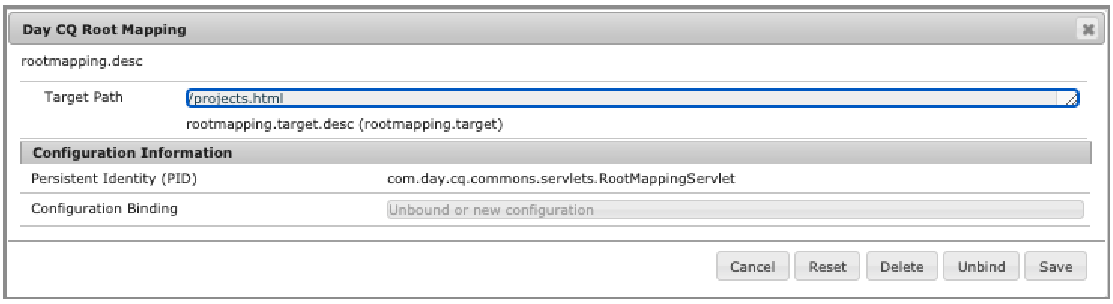
```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for devops 1.0-SNAPSHOT:
[INFO]
[INFO] devops ............................................. SUCCESS [  0.313 s]
[INFO] DevOps Project - Core .............................. SUCCESS [  7.499 s]
[INFO] DevOps Project - Repository Structure Package ...... SUCCESS [  0.736 s]
[INFO] DevOps Project - UI apps ........................... SUCCESS [  9.348 s]
[INFO] DevOps Project - UI content ........................ SUCCESS [  3.703 s]
[INFO] DevOps Project - All ............................... SUCCESS [  7.862 s]
[INFO] DevOps Project - Integration Tests Bundles ......... SUCCESS [  1.137 s]
[INFO] DevOps Project - Integration Tests Launcher ........ SUCCESS [  2.129 s]
[INFO] DevOps Project - Dispatcher ........................ SUCCESS [  0.456 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  34.655 s
[INFO] Finished at: 2020-06-03T13:59:44-07:00
[INFO] ------------------------------------------------------------------------
```

Note: Any changes you sync back to your maven project should be committed into source control:

git commit -a -m "added new files".

4. On a browser window, from the Navigation page, navigate to **Tools** > **Operations** > **Web Console** to open the Web Console Config Manager.

5. Search on the web page for **Day CQ Root Mapping**, as shown:



Day CQ Root Mapping

rootmapping.desc

Target Path: /projects.html

rootmapping.target.desc (rootmapping.target)

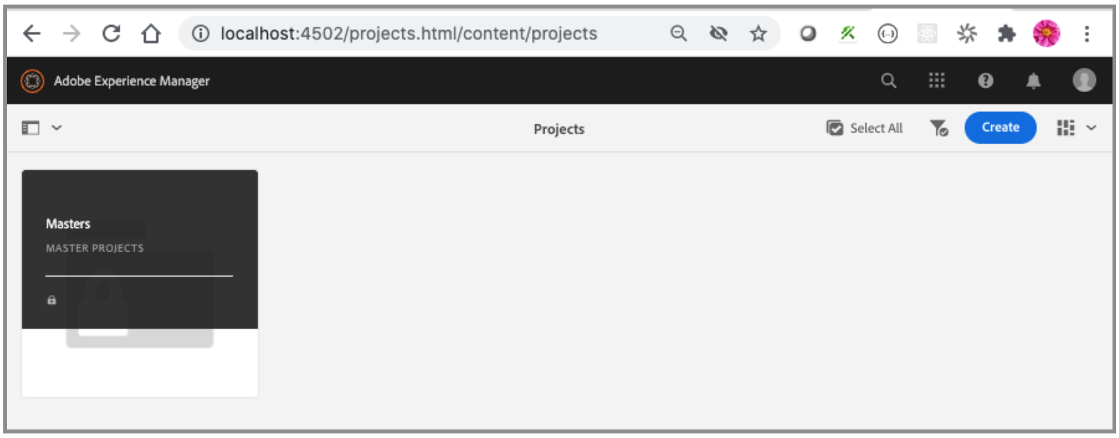
Configuration Information

Persistent Identity (PID): com.day.cq.commons.servlets.RootMappingServlet

Configuration Binding: Unbound or new configuration

The Target Path value should be /projects.html. This value matches the Day CQ Root Mapping configuration stored in the config.author.dev folder.

6. On your browser and navigate to http://localhost.
   The root access should redirect to http://localhost:4502/projects.html/content/projects, as shown:



localhost:4502/projects.html/content/projects

You now know how to create different OSGi configurations for specific runmodes.

ACS Commons: An Overview

Originally created by the Adobe Consulting Services group, AEM commons includes a broad range of useful items that fill some gaps in the out-of-the-box implementation of AEM. From servlets for capturing a series of image transforms under a re-usable label, to common workflow processes, and the tools to expand the logging capabilities of your AEM instance, AEM commons addresses many missing features.

Note: To learn more about ACS Commons, look at [1].

Exercise 3: Install ACS Commons for the dev environment

Scenario: A developer is interested in exporting the AEM taxonomy as a CSV file. The developer searched for a tool to make the job easier as the developer posited that other customers would have found this functionality useful. The developer found the Tag to CSV Exporter tool in ACS Commons.

Prerequisites:

- AEM SDK
- Local AEM author service running with dev run mode
- AEM project created from Maven AEM Project Archetype
- AEM project imported into Eclipse
- Maven 'Run As" Profiles defined

This exercise includes the following tasks:

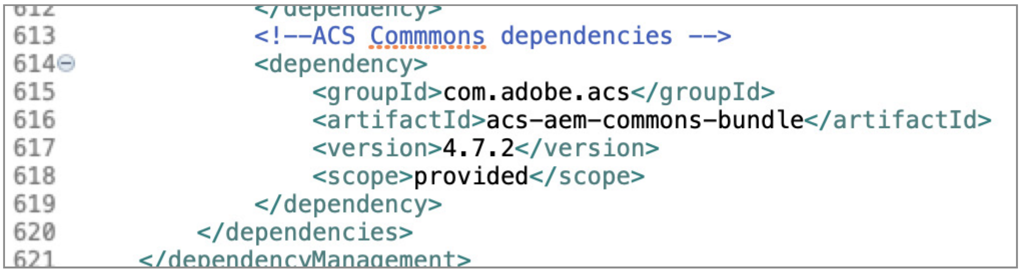
1.Include ACS project configuration and dependencies into the AEM project pom files

2.Install and validate ACS Commons

Task 1: Include ACS project configuration and dependencies into the AEM project pom files

1. Using a file browser, navigate to **Exercise_Files/training-files/RunModes**. Open the **updated-parent-pom.xml** file.

2. Find the dependency declaration for ACS Commons, as shown:

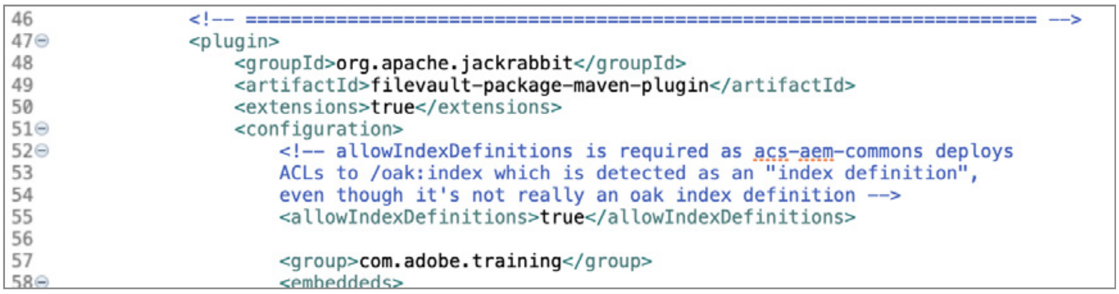


```
        </dependency>
        <!--ACS Commons dependencies -->
        <dependency>
            <groupId>com.adobe.acs</groupId>
            <artifactId>acs-aem-commons-bundle</artifactId>
            <version>4.7.2</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

3. Using Eclipse, replace the contents of the **devops > pom.xml** file with the contents of **updated-parent-pom.xml**.

4. Using a file browser, navigate to **Exercise_Files/training-files/RunModes**. Open the **updated-all-pom.xml** file.

5. Find the <allowIndexDefinitions> element, as shown:

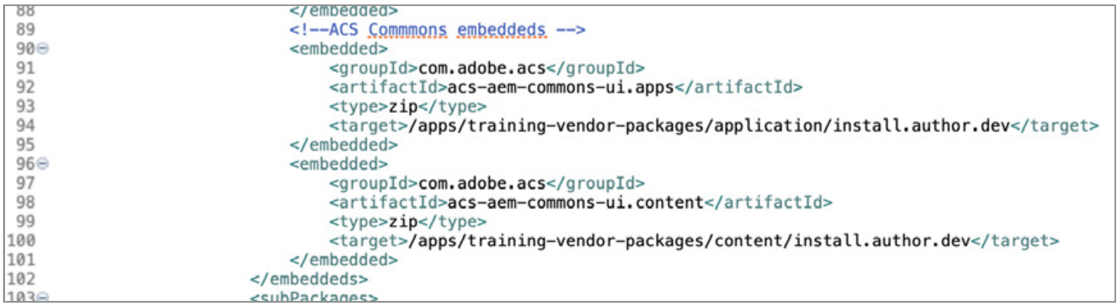


```
        <!-- ====================================================================== -->
        <plugin>
            <groupId>org.apache.jackrabbit</groupId>
            <artifactId>filevault-package-maven-plugin</artifactId>
            <extensions>true</extensions>
            <configuration>
                <!-- allowIndexDefinitions is required as acs-aem-commons deploys
                ACLs to /oak:index which is detected as an "index definition",
                even though it's not really an oak index definition -->
                <allowIndexDefinitions>true</allowIndexDefinitions>

                <group>com.adobe.training</group>
                <embeddeds>
```

Note: The element is required, as acs-am-commons deploys ACLs to /oak:index, which is detected as an "index definition", even though it's not really an oak index definition.

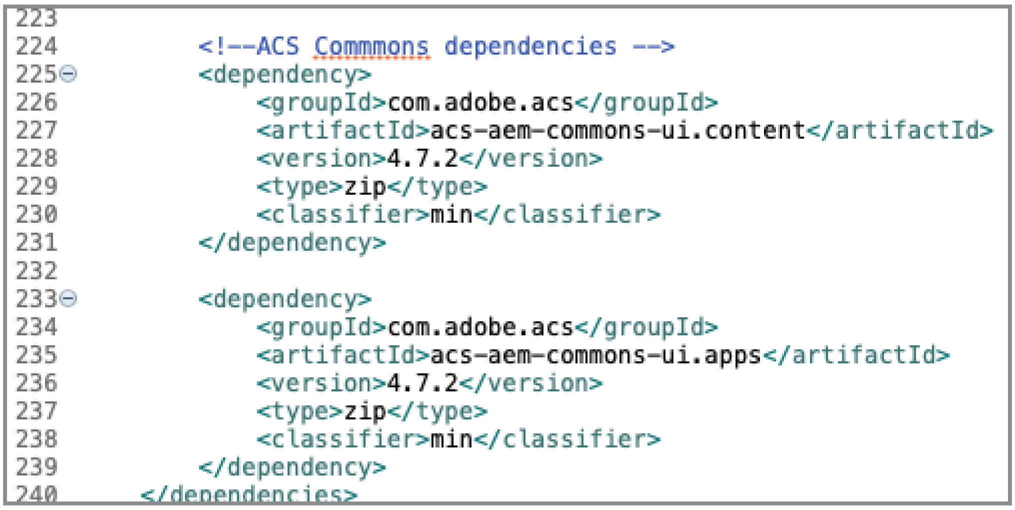
6. Find the <embedded> declarations for the ACS Commons application and content packages, as shown:



```
                    </embedded>
                    <!--ACS Commmons embeddeds -->
                    <embedded>
                        <groupId>com.adobe.acs</groupId>
                        <artifactId>acs-aem-commons-ui.apps</artifactId>
                        <type>zip</type>
                        <target>/apps/training-vendor-packages/application/install.author.dev</target>
                    </embedded>
                    <embedded>
                        <groupId>com.adobe.acs</groupId>
                        <artifactId>acs-aem-commons-ui.content</artifactId>
                        <type>zip</type>
                        <target>/apps/training-vendor-packages/content/install.author.dev</target>
                    </embedded>
                </embeddeds>
                <subPackages>
```

Notice the <target> path. This is where Maven will deploy the package.

7. Find the dependency declarations for ACS commons.



```
        <!--ACS Commmons dependencies -->
        <dependency>
            <groupId>com.adobe.acs</groupId>
            <artifactId>acs-aem-commons-ui.content</artifactId>
            <version>4.7.2</version>
            <type>zip</type>
            <classifier>min</classifier>
        </dependency>

        <dependency>
            <groupId>com.adobe.acs</groupId>
            <artifactId>acs-aem-commons-ui.apps</artifactId>
            <version>4.7.2</version>
            <type>zip</type>
            <classifier>min</classifier>
        </dependency>
    </dependencies>
```

Tip: ACS AEM Commons has two distributions

- The 'full' package that includes all functionality.
- The 'min' package that excludes functionality requiring third party dependencies.

Currently, the only feature excluded from the 'min' package is the Twitter integration.

To include the 'min' package, add

<classifier>min</classifier>

inside the <dependency> element.

To include the 'full' package, do not provide any <classifier> element inside the <dependency> element.

8. Using Eclipse, replace the contents of the **devops > all > pom.xml** file with the contents of **updated-all-pom.xml**.

At this point, you have configured the inclusion of the ACS Commons project into your project. Next step is to deploy to a local AEM Author instance running with the dev run mode.

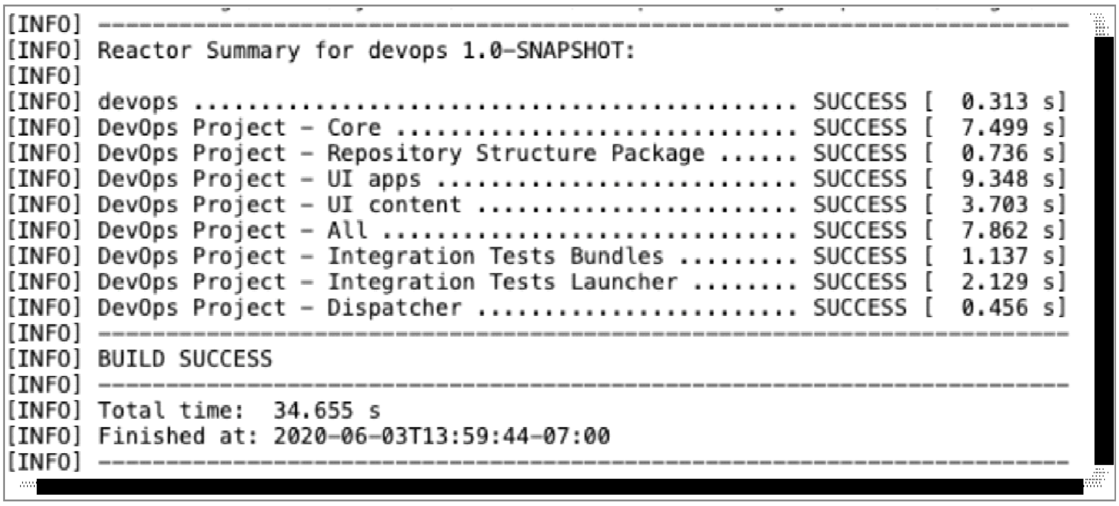
Task 2: Install and validate ACS Commons

1. If you do not already have a command line window open in your < AEM Project > directory, open one now.

2. Enter the following command to deploy your changes to the local AEM author instance:

```
$ mvn clean install -P adobe-public -P autoInstallSinglePackage
```

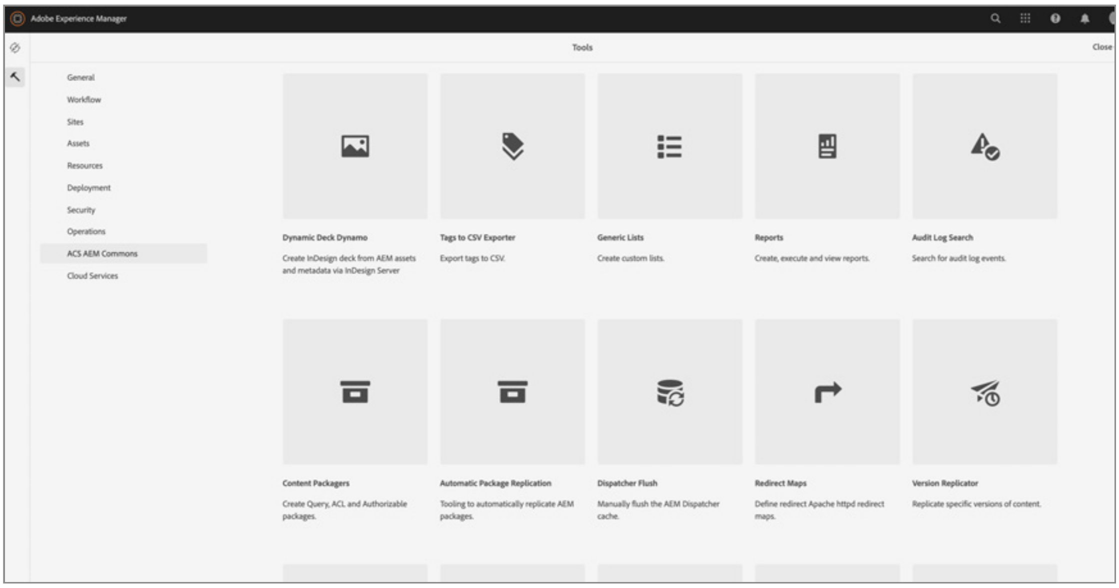
3. Verify that the build is completed successfully, as shown:



```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for devops 1.0-SNAPSHOT:
[INFO]
[INFO] devops ............................................. SUCCESS [  0.313 s]
[INFO] DevOps Project - Core .............................. SUCCESS [  7.499 s]
[INFO] DevOps Project - Repository Structure Package ...... SUCCESS [  0.736 s]
[INFO] DevOps Project - UI apps ........................... SUCCESS [  9.348 s]
[INFO] DevOps Project - UI content ........................ SUCCESS [  3.703 s]
[INFO] DevOps Project - All ............................... SUCCESS [  7.862 s]
[INFO] DevOps Project - Integration Tests Bundles ......... SUCCESS [  1.137 s]
[INFO] DevOps Project - Integration Tests Launcher ........ SUCCESS [  2.129 s]
[INFO] DevOps Project - Dispatcher ........................ SUCCESS [  0.456 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  34.655 s
[INFO] Finished at: 2020-06-03T13:59:44-07:00
[INFO] ------------------------------------------------------------------------
```

> **Note**: Any changes you sync back to your maven project should be committed into source control:
>
> **git commit -a -m "added new files"**.

4. From the Navigation page, navigate to **Tools** > **ACS Commons** to investigate the available ACS Commons functionality, as shown:



Dynamic Deck Dynamo
Create InDesign deck from AEM assets and metadata via InDesign Server

Tags to CSV Exporter
Export tags to CSV.

Generic Lists
Create custom lists.

Reports
Create, execute and view reports.

Audit Log Search
Search for audit log events.

Content Packagers
Create Query, ACL and Authorizable packages.

Automatic Package Replication
Tooling to automatically replicate AEM packages.

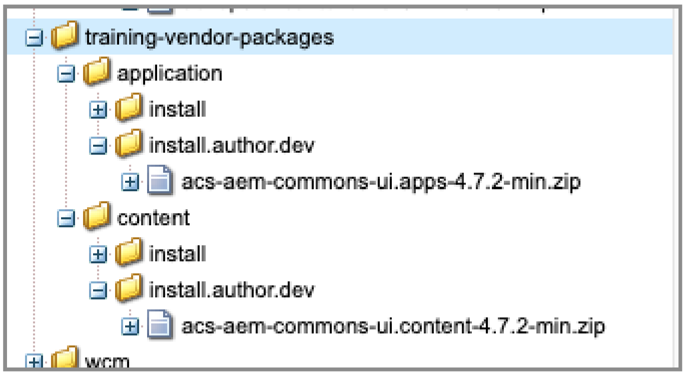
Dispatcher Flush
Manually flush the AEM Dispatcher cache.

Redirect Maps
Define redirect Apache httpd redirect maps.

Version Replicator
Replicate specific versions of content.

You can also see the deployed packages by using CRXDE Lite.

5. Using **CRXDE Lite**, navigate to **/apps/training-vendor-packages/content/install.author.dev**, as shown:

   ```
   training-vendor-packages
     application
       install
       install.author.dev
         acs-aem-commons-ui.apps-4.7.2-min.zip
     content
       install
       install.author.dev
         acs-aem-commons-ui.content-4.7.2-min.zip
   wcm
   ```

   Notice that the location of the ACS Commons packages matches the <embedded> module <target> path that was defined in the all project *pom.xml* file.

You now know the value of ACS Commons. You have also learned how to include third party projects into your AEM as a Cloud Service application.

References

1. ACS Commons: https://adobe-consulting-services.github.io/acs-aem-commons/ ↩