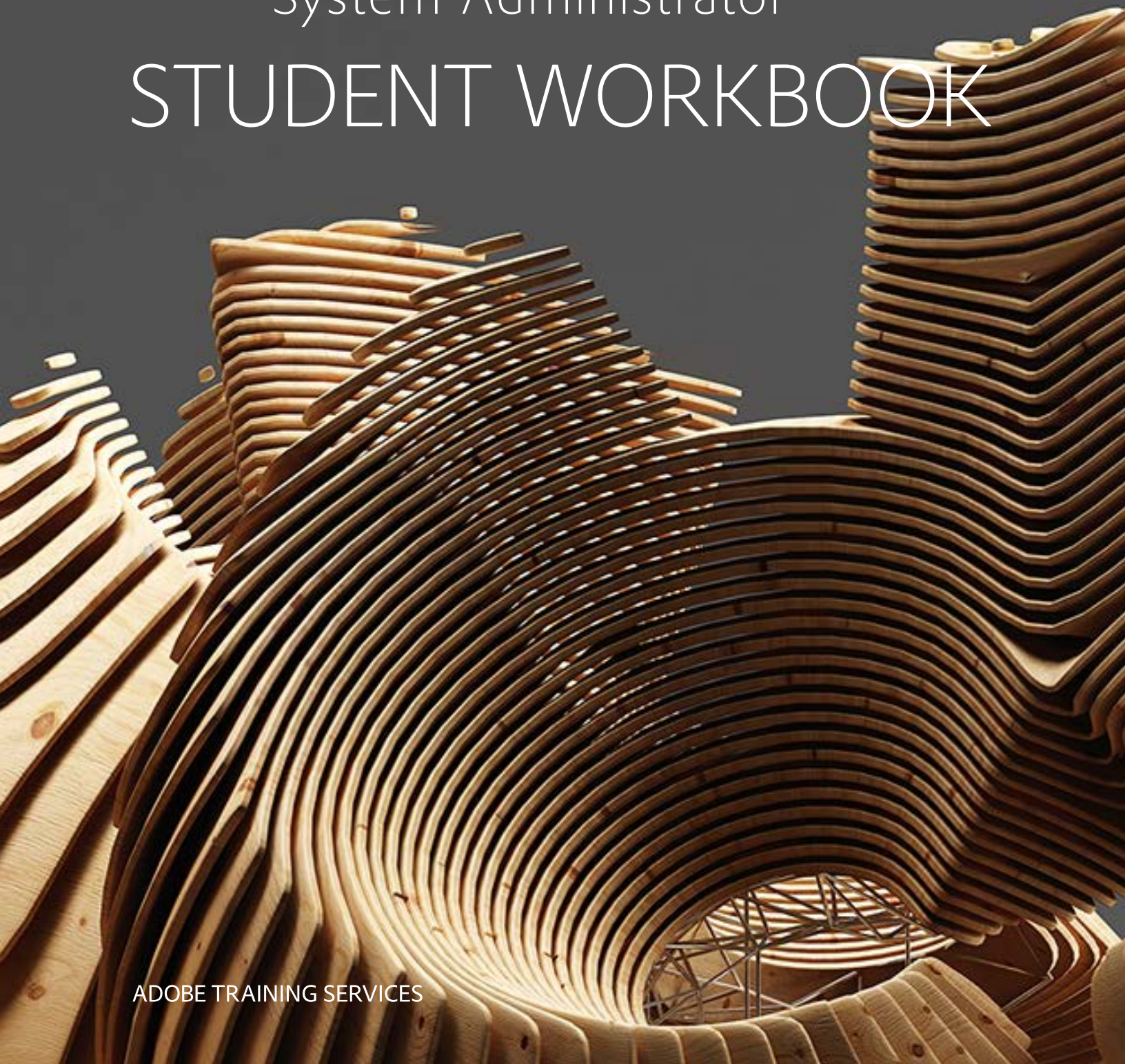
Adobe Experience Manager

AEM 6.x Sites: System Administrator

STUDENT WORKBOOK



ADOBE TRAINING SERVICES

Adobe Experience Manager 6.1: System Administrator

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

28MAY2015

Table of Contents

Chapter One	Introduction, Install & Start an Author Instance	1-1
	Overview	1-1
	OSGi and Apache Sling	1-1
	Objective	1-1
	AEM Functional Building Blocks	1-2
	Granite Platform	1-2
	Architecture Stack	1-3
	OSGi Bundles	1-5
	Additional Information	1-5
	OSGi Framework	1-6
	JCR	1-7
	JCR Structure	1-8
	Content Services of the JCR	1-8
	Apache Jackrabbit	1-9
	Apache Jackrabbit Oak	1-9
	Oak Architecture (Also Known as Hamburger Architecture)	1-10
	Oak Versus CRX	1-11
	Microkernels	1-11
	Adobe CRX	1-12
	Built-in Protocols/APIs for the CRX Platform	1-12
	Repository Structure	1-12
	Install and Start an AEM Instance	1-14
	What Is an Author Instance?	1-15
	Exercise 1.1 Install an AEM Author Instance	1-15
	Start the Author Instance	1-16
	Graphical Start	1-16
	After AEM Has Started	1-16
	Command Line Start	1-17
Chapter Two	UI Introduction and Page Actions	2-1
	Overview	2-1
	Objective	2-1
	The AEM UIs	2-2
	AEM Welcome Screen Utilities	2-3
	Websites Console (Classic Interface)	2-4
	Websites Console (Touch-Optimized UI)	2-4

	Accessing the Classic UI	2-9
	Tools Console	2-10
	Explore the Touch UI for Available Admin Tools	2-10
	Exercise 2.1 Edit a Page in the Touch UI	2-12
Chapter Three	Exploring the Administrative Interfaces	3-1
	Overview	3-1
	Objectives	3-1
	What Interfaces Exist?	3-2
	Exercise 3.1 Use the Adobe Web Console	3-2
	Exercise 3.2 Use CRXDE Lite	3-3
Chapter Four	Using the CRX Package Manager and Automating it With cURL	4-1
	Objective	4-1
	Why Do You Need CRX Content Packages?	4-2
	Exercise 4.1 Create, Build, and Download a CRX Package	4-2
	Configuring cURL for Windows	4-6
	Installing cURL in Mac OS X	4-6
	Exercise 4.2 Use Package Manager with cURL	4-7
Chapter Five	Configuring OSGi Bundles, Log files, and Setting Runmodes	5-1
	Overview	5-1
	Resolution Order at Startup	5-2
	Structure Within the Repository	5-2
	Version Manager	5-3
	Exercise 5.1 Configure the Version Manager	5-5
	Exercise 5.2 Create a New Version Using the Touch-Optimized UI	5-8
Chapter Six	Custom Log Files	6-1
	Goal	6-1
	Exercise 6.1 Create a Custom Log File	6-2
	Create the Logging Writer	6-3
	Run Modes	6-4
	Setting Run Modes	6-5
	Configurations Per Run Mode	6-5
	Configurations for Different Run Modes	6-5
	Additional Information on Run Modes	6-5
	Using Custom Run Modes With Configurations	6-6
	Create a CRX Logger	6-8
Chapter Seven	Replication and Reverse Replication Agents	7-1
	Overview	7-1
	Set Up the Replication Agent for Two Publish Instances	7-1
	AEM Environment Basic Setup	7-2
	Replication Agents	7-2
	Transport Tab Configuration Parameters	7-4
	Proxy Tab Configuration Parameters	7-5
	Extended Tab Configuration Parameters:	7-6
	Triggers Tab Configuration Parameters	7-7
	Batch Tab Configuration Parameters	7-8
	How to Monitor Replication Agents	7-8
	Exercise 7.1 Install Two Publish Instances to Configure Replication	7-10

	Publish Instance Folder Structure	7-10
	Exercise 7.2 Configure and Access Replication Agents	7-11
	Exercise 7.3 Activate a Tree	7-14
	Reverse Replication Agent	7-15
	Exercise 7.4 Add a Reverse Replication Agent	7-16
Chapter Eight	Dispatcher	8-1
	Goal	8-1
	AEM Dispatcher Module Overview	8-1
	Caching Methods	8-2
	Caching Choices	8-2
	Dispatcher Module Architecture	8-2
	Dispatcher Caching Algorithm	8-3
	AEM Dispatcher and Load Balancing	8-3
Chapter Nine	Add Dispatcher to the Apache Web Server	9-1
	Install Apache	9-2
	Exercise 9.1 Install the Apache Web Server	9-2
	Install the Dispatcher Module	9-2
	Exercise 9.2 Install Dispatcher Into the Apache Web Server	9-2
	Configure Apache	9-3
	Exercise 9.3 Configure Apache Web Server	9-3
Chapter Ten	Configure Dispatcher	10-1
	Goal	10-1
	Understanding the dispatcher.any File contents	10-5
	Exercise 10.1 Configure the dispatcher.any File	10-8
	Exploring more of dispatcher.any File contents	10-9
	Exercise 10.2 Complete the Dispatcher Configuration	10-11
	Exercise 10.3 Configure the Dispatcher Flush Agent	10-12
	Testing the Dispatcher Flush Agent	10-13
	Full Dispatcher Test	10-14
Chapter Eleven	Oak Queries and Indexing	11-1
	Objectives	11-1
	JCR/JSR-283	11-1
	Goals	11-1
	Microkernels and NodeStores	11-2
	DocumentStore	11-3
	SegmentStore	11-4
	Oak Queries and Indexing	11-4
	Oak Query Implementation Overview	11-5
	Native Queries	11-6
	Indexer Types	11-6
	Indexing Tools	11-8
Chapter Twelve	Introduction to MongoDB and Relational Database MK for Oak	12-1
	Objectives	12-1
	Exercise 12.1 Install AEM With MongoDB	12-2
	What Are mongo Shell, mongod, and mongos for MongoDB?	12-6
	MongoDB Microkernel (MongoDocumentStore)	12-6
	Relational Database MK for Oak	12-6

Chapter Thirteen	Recommended Deployments and Scaling AEM With MongoDB	13-1
	Recommended Deployments	13-1
	Deployment Scenarios	13-1
	Microkernels: Which One to Use	13-6
	Choosing the deployment type for Social Communities	13-7
	MongoMK with Active-Active Cluster	13-8
	Exercise 13.1 Setup Mongo Clusters	13-9
	Exercise 13.2 Configure AEM 6.1 Instances With MongoDB Cluster	13-10
	Exercise 13.3 Validate the AEM/MongoDB Cluster	13-11
	Managing CRX3 Clusters	13-12
	FAQ About Sharding and Replica Sets	13-12
	Exercise 13.4 Explore AEM 6.1 with MongoDB	13-13
Chapter Fourteen	Backing Up AEM 6.1	14-1
	Purpose of Backup	14-1
	TarMK Backup	14-1
	MongoMK Backup	14-1
	DataStore Backup (Binaries)	14-2
	Exercise 14.1 Create a Backup	14-5
	Backing Up the DataStore Separately	14-5
	Backing Up to the Default Target Directory	14-6
	Backing Up to a Non-default Target Directory	14-6
	Backing Up a Large Repository	14-6
Chapter Fifteen	User Administration and Security	15-1
	Goal	15-1
	Users and Groups	15-2
	Accessing User Administration With the Security Console	15-2
	Exercise 15.1 Create a New User	15-4
	Exercise 15.2 Create a Group	15-5
	Exercise 15.3 Add a User and Groups to a Group	15-6
	Exercise 15.4 Impersonate a User	15-7
	Exercise 15.5 Manage Permissions	15-8
	Permissions and ACLs	15-9
	Actions	15-10
	ACLs and How They Are Evaluated	15-10
	Concurrent Permission on ACLs	15-12
	Exercise 15.6 Manage Access Rights for Different Websites	15-13
	Exercise 15.7 Manage Replication Privileges	15-13
	Exercise 15.8 Deny Access Rights to Consoles	15-14
	Remove Access to the Navigation Option in the Rail	15-15
	Exercise 15.9 Manage Conflicting ACLs	15-16
	Exercise 15.10 Delete Users or Groups	15-24
Chapter Sixteen	Integrating With LDAP and Enabling Single Sign On	16-1
	Goal	16-1
	Configuring the LDAP Identity	16-2
	Provider Configuring the Synchronization Handler	16-3
	The External Login Module	16-4
	Exercise 16.1 Install a Local LDAP Server	16-5
	Exercise 16.2 Set Up/Configure the LDAP Server	16-6

Exercise 16.3 Configure AEM to Integrate With LDAP 16-10
Exercise 16.4 Assign Rights to Users Imported From LDAP Into CRX 16-12
Purge Users in LDAP 16-13
Exercise 16.5 Purge User007 16-13
Synchronize the CRX With LDAP 16-16
Exercise 16.6 Synchronize the CRX With LDAP 16-16
Removing AEM Sign-Out Links 16-21
Optional Exercise 16.7 Configure LDAP Over SSL 16-22
Optional Exercise 16.8 Create SSL Certificates 16-22
Enabling Debug Logging 16-23

Chapter Seventeen Performance Tuning 17-1
Goal 17-1
What Performance Optimization Concepts Should I Consider? 17-2
Performance Optimization Methodology 17-2
Planning for Optimization 17-2
Simulate Reality 17-3
Establish Solid Goals 17-3
Stay Relevant 17-3
Agile Iteration Cycles 17-3
Basic Performance Guidelines 17-3
Exercise 17.1 Monitor Page Response 17-4
Exercise 17.2 Find the Response Performance 17-5
Exercise 17.3 Monitor Component-based Timing 17-5

Chapter Eighteen Security Checklists 18-1
CRX Security Checklist 18-1
Issues with Cross-Site Request Forgery 18-1
Sanity Checks Prior to Go-Live 18-2
Exercise 18.1 Disable WebDAV and Debug Mode 18-4
Disable Debug Mode 18-4
Production-Ready Lockdown Package in AEM 6.1 18-5

Chapter Nineteen Existing System Console and New Dashboard 19-1
Operations Dashboard 19-1
Audit Log Maintenance in AEM 6.1 19-2
Exercise 19.1 Work With System Checks 19-2
Exercise 19.2 Work With Active Bundles 19-8
Exercise 19.3 Work With Security Checks 19-10
Monitoring 19-12
External Login Modules 19-13
LDAP Configuration 19-14
Use JMX and the MBeans Provided by CRX 19-14

Appendix-A Configuring the AEM Dispatcher on Mac OS X A-1

Appendix-B Dispatcher Installation on a Different Version of Microsoft IIS B-1

Appendix-C Running a MongoDB Instance, Clustering in AEM Upgrade/Migration to AEM 6.0 C-1

Introduction, Install & Start an Author Instance

Overview

This chapter gives an introduction to the OSGi, Apache Sling frameworks, and clustering.

OSGi and Apache Sling

Adobe Experience Manager (AEM) is built within an OSGi application framework. OSGi is a dynamic module system for Java that provides a framework within which small, reusable, standardized components can be composed into an application and deployed. The Apache Sling framework is designed to expose a Java Content Repository (JCR) through an HTTP-based Representational State Transfer (REST) API. Both AEM's native functionality and the functionality of any website built with AEM are delivered through this framework.

Objective

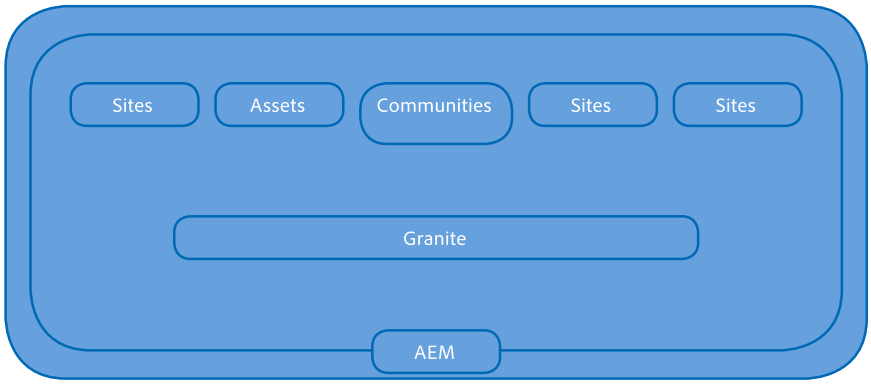
In this chapter, you will learn about the following:

- OSGi and Apache Sling
- AEM functional building blocks
- Granite platform
- OSGi framework
- OSGi bundles

AEM Functional Building Blocks

AEM consists of sets of OSGi bundles that are deployed into the CRX. The AEM application modules use the JCR API to manipulate content in CRX. The AEM application modules sit on top of the AEM shared framework, which contains all application layer functionality that is shared among all the application modules, for example, mobile functionality, multi-site manager, taxonomy management, and workflow. In addition to the shared application framework, the AEM applications (sites, assets, and so on) share the same infrastructure and UI framework. With the installation of AEM, you receive all applications because they are tightly integrated. Application functionality that is not used or licensed can be disabled after the initial installation.

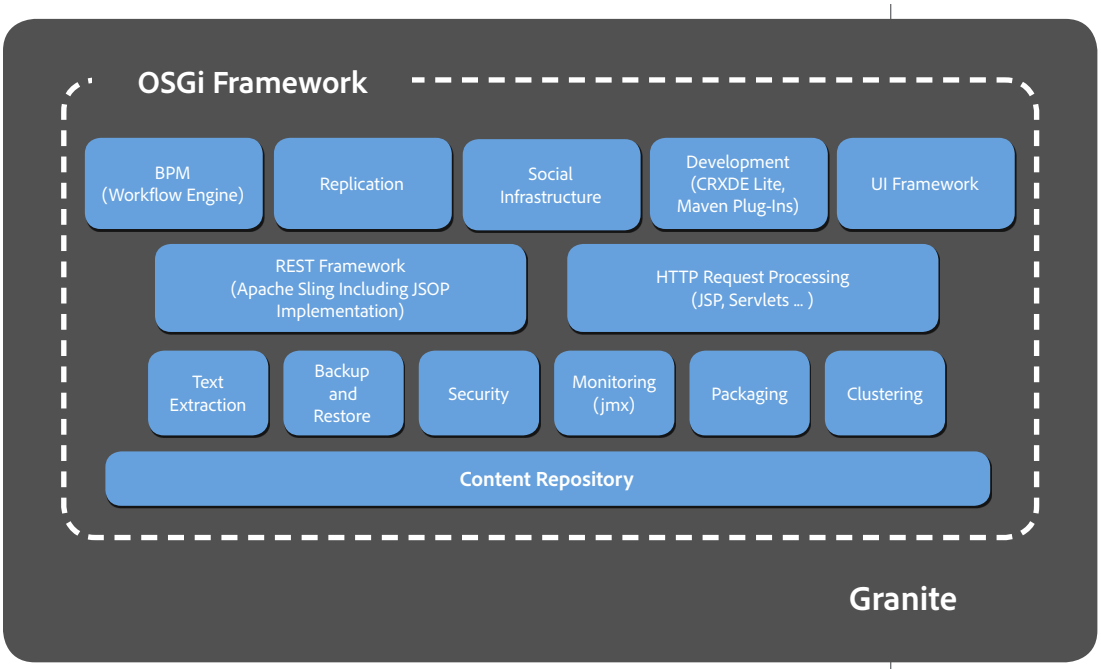
Third-party repositories can be integrated with JCR connectors that expose their content into CRX and are available to AEM. Notice that the connectors are plugged in at the content repository level, which allows the content in the external repositories to appear to authors as if the content existed in the local content repository—a true virtual repository.



Granite Platform

The AEM application modules sit on top of the AEM shared framework (granite), which contains functionality that is shared among all the application modules, for example, mobile functionality, multi-site manager, taxonomy management, and workflow.

In addition to the shared application framework, the AEM applications (Web Content Management (WCM), DAM, Mobile, and so on) share the same infrastructure and same UI framework.
Because 'Everything Is Content' and all the content is in the content repository, you will note that clustering and backup are done at the repository level.



AEM is implemented as a Java web application. Because the system can be accessed from any computer with a modern web browser, no installation of client software is required. Therefore, large installations with thousands of users can be rolled out and upgraded easily.

The Application Runtime is OSGi, specifically Apache Felix. At this layer, you can hot deploy any code that you wrap up as an OSGi bundle. The OSGi runtime hosts Java applications that can access the repository through the JCR API.

As part of the Application Runtime, you receive Apache Sling, a RESTful web application framework that exposes the full repository content through HTTP and other protocols.

Architecture Stack

AEM Architectural Stack

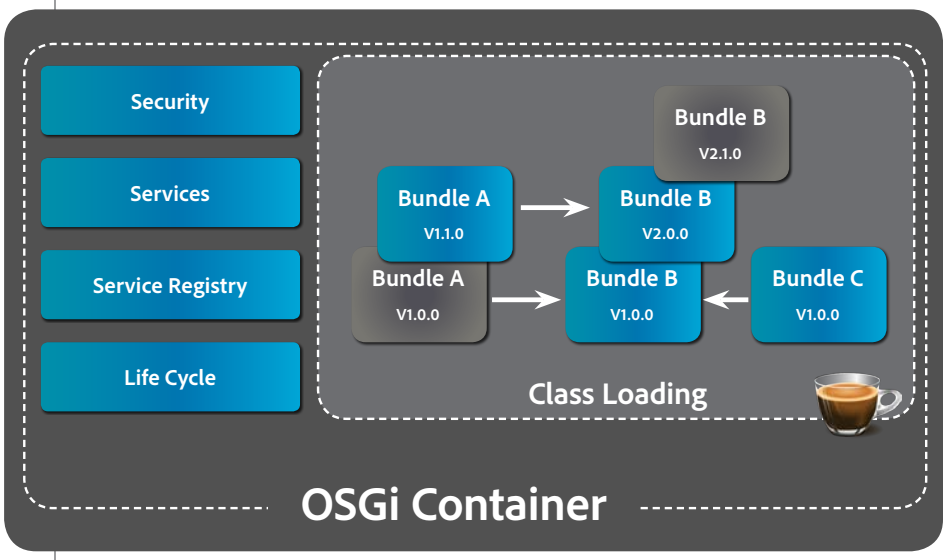
OSGi is a consortium that has developed a specification to build modular and extensible applications. The OSGi module system allows building applications as a set of reloadable and strongly encapsulated services.

Traditional Java application servers use a monolithic approach to application deployment. OSGi bundles run inside an OSGi container. The OSGi container manages relations among bundles, which are JAR files that contain extra metadata indicating what services they require and provide. The OSGi specifications define a dynamic component system for Java:

- OSGi specifications enable:
  - A development model where applications are (dynamically) composed of many different (reusable) components
  - Components to hide their implementations from other components while communicating through services, which are objects specifically shared between components
- Collaborative software environment
  - The application emerges from assembling multiple, reusable modules that have no previous knowledge of each other.

OSGi is a solution to the new DLL-hell, which is the Java class loader. You can have more than one version of any bundle running in the container at the same time. The container resolves any version dependencies.

OSGi is a platform in the Java stack that allows large development teams to be more efficient and provides the ability to replace code pieces (bundles) during normal operations of the application—in other words, without taking the server down.



The OSGi container (Apache Felix) that is included in CRX is compliant to version 4.2 of the OSGi specification.

Benefits of using OSGi solution:

- Easy to write and test the code
- Increased reuse of components
- Simple build systems
- Manageable deployment
- Early detection of bugs
- Runtime provides an enormous insight into what is running

OSGi Bundles

- OSGi bundles can contain compiled Java code, scripts, and content that is to be loaded into the repository, in addition to configuration and/or other files, as needed.
- Bundles can be loaded and installed during normal operations.
- In AEM, bundles are dropped into specially named folders (.../install) in the repository. The Apache Felix Management Console can also manage them.

Additional Information

To learn more about OSGi, go to:

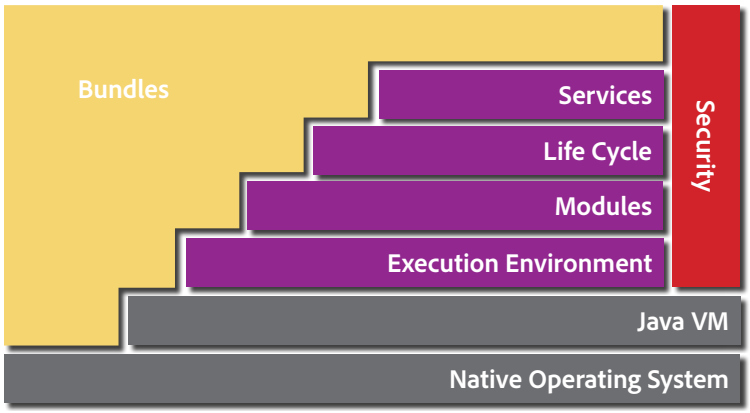
- http://www.osgi.org
- http://en.wikipedia.org/wiki/OSGi

To learn more about Apache Felix specifically, go to:

- felix.apache.org

OSGi Framework

The OSGi framework is made up of three layers—Module, Life Cycle, and Services—that define how extensible applications are built and deployed.



The responsibilities of the layers are:

- **Module:** The module layer defines how a module or bundle in OSGi-speak is defined. A bundle is a JAR file whose manifest file has some defined entries. These entries identify the bundle with a symbolic name, a version, and so on. In addition, there are headers that define what a bundle provides— Export-Package; and what a bundle requires to be operative—Import-Package, and Require-Bundle.
- **Life Cycle:** The life cycle layer defines the states a bundle may be in and describes the state changes. By providing a class, which implements the Bundle-Activator interface and which is named in the Bundle-Activator manifest header, a bundle may hook, into the life cycle process when the bundle is started and stopped.
- **Services:** For the application to be able to interact, the OSGi Core specification defines the service layer. This describes a registry for services, which may be shared.

The key principles of OSGi are:

- Universal middleware, dynamic module system
- Service-oriented, component-based environment
- Standardized software life cycle management

The implementation of OSGi that the AEM platform makes use of is Apache Felix.

JCR

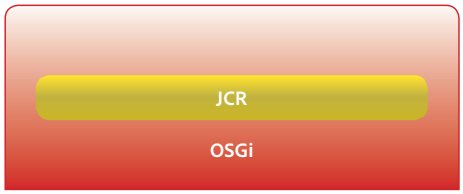
According to JSR-283, the JCR API defines an abstract model and a Java API for data storage and related services commonly used by content-oriented applications.

A JCR is an object database that provides various services for storing, accessing, and managing content. In addition to a hierarchically structured storage, common services of a content repository are versioning, access control, full text searching, and event monitoring.

The JCR provides a generic application DataStore for structured and unstructured content. File systems provide excellent storage for unstructured, hierarchical content. Databases provide excellent storage for structured data due to transactional services and referential integrity functionality. The JCR provides the best of both data storage architectures, observation, versioning, and full text search.

An additional advantage of the JCR is support for namespaces. Namespaces prevent naming collisions among items and node types that come from different sources and application domains. JCR namespaces are defined with a prefix, delimited by a single colon (:); for example, jcr:title.

AEM is designed to store and retrieve content from any JCR-compliant content repository. In its default configuration, content storage is handled by the CRX repository that comes bundled with AEM. CRX is Adobe's implementation of the JCR standard. Oak is an effort to implement a scalable and efficient hierarchical content repository for use as the foundation of modern world-class web sites and other demanding content applications. It is the successor to Jackrabbit 2 and is used by AEM 6.1 as the default backend for its content repository, CRX.



In addition to CRX, AEM can also work with other JCR repositories such as Apache Jackrabbit and with a number of non-JCR DataStores through connectors. As AEM is built on top of this standard, it is capable of pulling content not just from its built-in CRX repository, but also from any JCR-compliant source, such as a third-party repository (for example, Jackrabbit) or a connector that exposes legacy storage through JCR.

JCR defines a Java API for a class of data storage systems called content repositories. A content repository, as defined by JCR, combines features of the traditional, relational database with those of a conventional file system, as well as additional services that content-centric applications often need—but that neither file systems nor databases typically provide. See the CRX and JCR documentation online for more information about this topic.

JCR Structure

The JCR consists of a set of one or more workspaces, each containing a tree of nodes with associated properties. Each node may have one primary node type that defines the characteristics of the node. They may also be associated with any node, zero or more mixins, which define additional node characteristics and behaviors.

Beginning with the root node at the top, the hierarchy descends much like the directory structure of a file system. Each node can have zero or more child nodes and zero or more properties. Properties cannot have children but do have values.

The values of properties are where the actual pieces of data are stored. These can be of different types—strings, dates, numbers, binaries, and so on. The structure of nodes above the properties serves to organize this data according to principles that the application using the repository employs.

Nodes may point to other nodes through a special reference type property. In this way, nodes in a JCR offer referential integrity and an object-oriented concept of inheritance.

Content Services of the JCR

- Search
- Indexing
- Observation
- Versioning
- Access control/security
- Transactions

Key principles behind the specification of JSR-283 are:

- A common programmatic interface to content repositories
- An API not tied directly to the underlying architecture, data source, or protocol
- Content organization in the repository model
- Hierarchical modeling

The reference implementation for JSR-283 is Apache Jackrabbit. To learn more about Apache Jackrabbit, go to jackrabbit.apache.org.
The Adobe implementation of JSR 283 is the CRX.

Apache Jackrabbit

The Apache Jackrabbit content repository is a fully conforming implementation of the content repository for Java Technology API (JCR, specified in JSR-170 and 283).

A content repository is a hierarchical content store with support for structured and unstructured content, full text search, versioning, transactions, and observation.

Apache Jackrabbit Oak

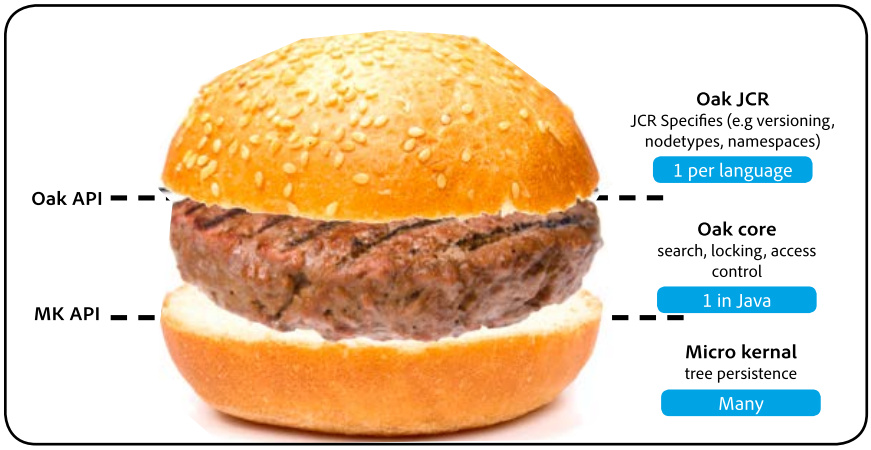
Jackrabbit Oak is an effort to implement a scalable and efficient hierarchical content repository for use as the foundation of modern, efficient web sites and other demanding content applications.

Jackrabbit Oak implements the JCR spec. The most used parts of JSR-283 are implemented, but Jackrabbit Oak does not aim to be a reference implementation of JSR-283. AEM 6.1 is built on top of Jackrabbit Oak. The following are the goals of Jackrabbit Oak:

- Scalability
  - Big repositories
  - Distributed repository with many cluster nodes
- Improved write throughput
  - Parallel writes
- Support for many child nodes
- Support for many Access Control Lists (ACLs)

Oak Architecture (Also Known as Hamburger Architecture)

The repository should implement standards like JCR, WebDAV, and Content Management Interoperability Services (CMIS), and be easily accessible from various platforms, especially from JavaScript clients running in modern browser environments. The implementation should provide more out-of-the-box functionality than typical NoSQL databases while achieving comparable levels of scalability and performance.



Architecture Concepts – Hamburger Architecture

The burger's top bun: Oak JCR

- Implements the JCR API
- Now implemented for JCR — Non-Java implementations possible and part of the concept

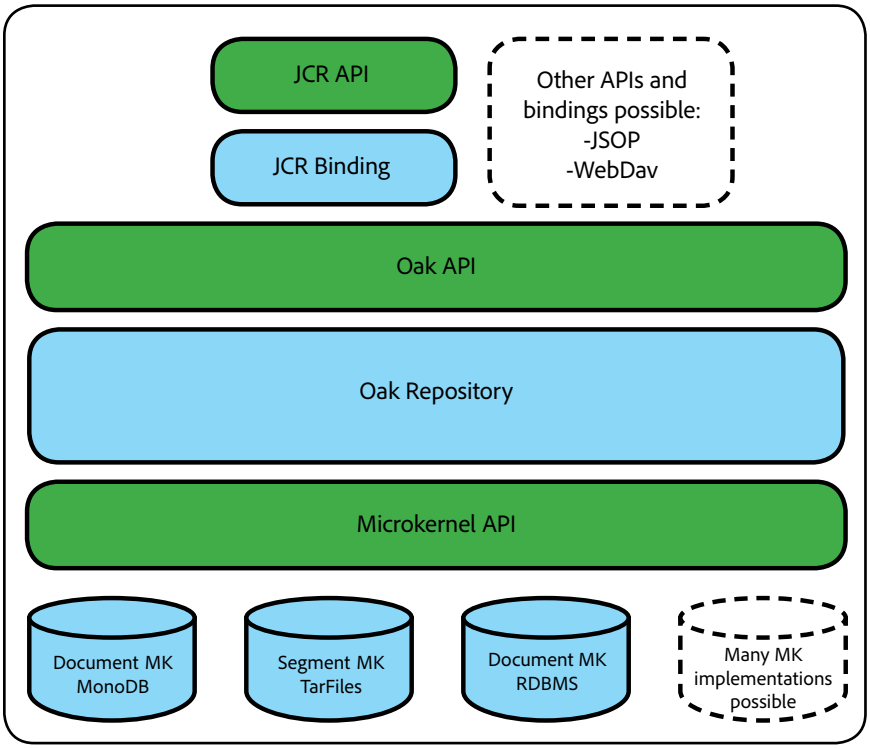
The burger's patty: Oak Core

- Where most of the heavy lifting takes place
- Adds to the MK's tree model (ACLs, search, and indexing)
- Observation
- Exposes a decorated tree model
- Mostly transforms JCR semantics into tree operations
- Also contains 'commit hooks' that implement JCR constraints; for example, node types

The burger's bottom bun: Microkernel

- Implements a tree model (nodes and properties)
- Exposes the Microkernel API

A more technical view of the Oak architecture: The architecture is designed to be extensible to offer different API and bindings on one hand and different Microkernel backend implementations on the other hand. Oak core is not interchangeable.



Oak Versus CRX

AEM 6.1 comes with the new JCR repository implementation based on Oak. This is enabled by default. Though there are differences in the way the new repository works internally, the applications running on AEM work in a similar manner. This is also applicable to Sling.

Microkernels

The Oak Microkernel API provides an abstraction layer for the actual storage of the content.

The Microkernel API is completely based on Strings. It uses the JSOP format, which is a lightweight HTTP protocol for manipulating JSON-based object models.

Currently, Oak has two main Microkernel implementations:

- DocumentMK
- SegmentMK

AEM 6.1 includes more options for DocumentMK such as DB2 and Oracle 12c. AEM 6.1 also supports MySQL, MariaDB, and Microsoft SQL Server on experimental support.

Adobe CRX

CRX implements the Content Repository API for Java Technology (JCR). This standard defines a data model and application programming interface (that is, a set of commands) for content repositories. Content is available through a standardized API:

- JSR 283
- JCR API
  - Node node.addNode("JCR")
  - Node node.getNode("JCR")
  - Node node.setProperty("opinion","excellent and recommended")
  - Node node.getProperty("opinion").getString()

Built-in Protocols/APIs for the CRX Platform

Add, consume, and manage content with these interfaces:

- Java Content Repository API: Complete JCR 2.0 implementation
- Content Management Interoperability Services: CMIS 1.0
- WebDAV: With versioning, access control, and search
- Windows Network File Share: CIFS/SMB
- RESTful Web API for JavaScript and Flash/Flex
- Java remoting with RMI and HTTP
- Lightweight Directory Access Protocol (LDAP) and any JAAS plug-in
- Native repository interface through the Virtual Repository: For example, Microsoft SharePoint

Repository Structure

You can use CRXDE Lite to explore the logical structure of the repository as defined by the JCR API. This interface is going to be useful to you as a developer. You will be able to look at the structures that your code writes into the repository.

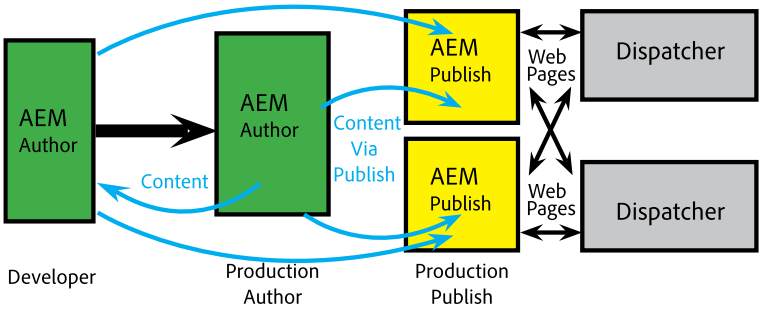
If you take a close look at the repository structure, you will notice that the repository has several major areas:

- **/var:** Files that change and are updated by the system, such as audit logs, statistics, and event handling
- **/libs:** Libraries and definitions that belong to the core of AEM. The sub-folders in /libs represent the out-of-the-box AEM features, such as search or replication. The content in /libs should not be modified because it affects the way AEM works. Features specific to your website should be developed under /apps.
- **/etc:** Utilities and tools. See the documentation for the Tools Console for detailed information.
- **/apps:** Application related. The custom templates and component definitions specific to your website. The components that you develop may be based on out-of-the-box components available at /libs/foundation/components.

- **/content:** Content created for your website
- **/tmp:** Temporary working area
- **/home:** User and group information

Install and Start an AEM Instance

The following structure shows the end-to-end workflow:



In AEM terminology, an 'instance' is a copy of AEM running on a server. AEM installations usually involve at least two instances, typically running on separate machines:

- **Author**: An AEM instance used to create, upload, and edit content, and to administer the website. After content is ready to go live, it is replicated to the Publish Instance.
- **Publish**: An AEM instance that serves the published content to the public

These instances are identical in terms of installed software. They are differentiated only by the configuration. In addition, most installations use a Dispatcher:

- **Dispatcher**: A static web server (Apache httpd, Microsoft IIS, and so on) augmented with the AEM Dispatcher module. It caches webpages produced by the Publish Instance to improve performance.

  There are many advanced options and elaborations of this setup, but the basic pattern of Author, Publish, and Dispatcher is at the core of most deployments. This section focuses on a relatively simple setup. The subsequent sections discuss advanced deployment options.

The following instructions explain how to install and start an Author instance. This is important because you will use this Author instance throughout this training to perform typical administration tasks. To successfully complete and understand these instructions, you will need:

- an AEM installation and startup JAR file (contained in the USB stick)

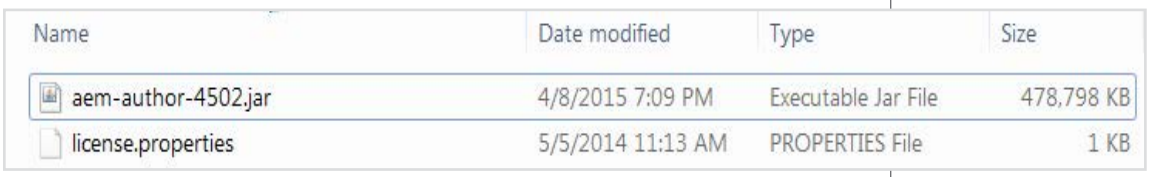
- a valid AEM license key 'properties' file (contained in the USB stick)
- a JDK version 1.7 or later
- approximately 4 GB of free space per instance
- approximately 4 GB of RAM

What Is an Author Instance?

There are two types of AEM instances. A Publish Instance is the slick and fast performing instance used by the Internet world (web). The Author instance typically resides behind a firewall and is used by content contributors (authors) to build and manage the website and its content. The Author instance is managed by a user-friendly GUI. Content authors will log in to the instance and use it to manage pages. This includes creating, editing, deleting, moving, and so on. In addition, the Author instance is the installation that developers will be developing against because you can easily observe Author and Publish views.

Exercise 1.1
Install an AEM Author Instance

1. Windows: Create a folder structure on your file system where you will store, install, and start AEM:
   a. `C:/adobe/AEM/author`
2. Mac OS or *x: Create the following structure commonly used for application installations:
   a. `/opt/adobe/AEM/author`
      OR
   b. `/Applications/AEM/author`
3. Copy the AEM quickstart JAR and license.properties file from <USB>/distribution/AEM_installation_file> into the newly created folder structure.
4. Rename the AEM quickstart JAR to cq-author-4502.jar.
   a. cq = the application
   b. author = the WCM mode it will run in (for example, Author or Publish)
   c. 4502 = the port it will run in (for example, any available port is acceptable)



Name	Date modified	Type	Size
aem-author-4502.jar	4/8/2015 7:09 PM	Executable Jar File	478,798 KB
license.properties	5/5/2014 11:13 AM	PROPERTIES File	1 KB

AEM Author Installation Folder Structure

NOTE: The AEM installation and startup JAR file is also known as the 'quickstart' file. It is used for installing AEM, and after AEM is installed, as the AEM startup file. You will notice during installation that the JAR file creates a root folder called 'crx-quickstart'.

NOTE: The AEM quickstart file is renamed for installation purposes. When running for the first time, the quickstart file will notice that it has to install AEM. By renaming the file, you use a convention of passing instance name (Webpathcontext) and port number through the file name so that no user interaction is needed during the installation process. If no port number is provided in the file name, AEM will select the first available port from the following list:
1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, random.

Start the Author Instance

There are two ways of installing the AEM—graphical and by command line. The latter is more powerful because the user has the possibility to provide additional performance-tuning parameters to the JVM. See the section corresponding to your desired installation method.

Graphical Start

In a Windows or Mac OS environment, you can double-click the cq-author-4502.jar file.

- Installation will take approximately 3-5 minutes, depending on your system's capabilities
- A dialog box will pop up similar to the following:

AEM Installation Startup Dialog Box

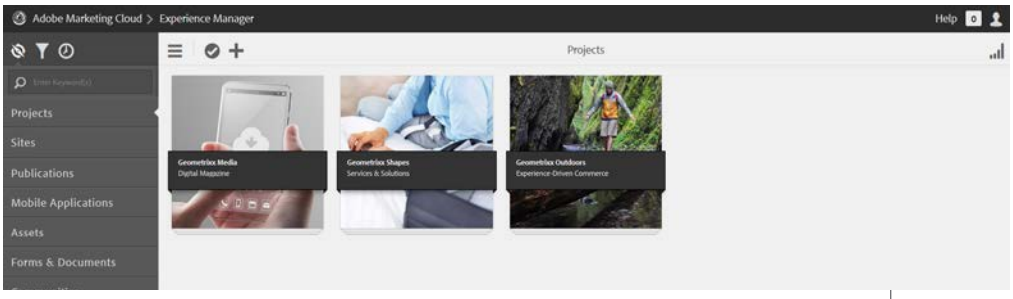
After AEM Has Started

After AEM has started, your default browser will be automatically opened, pointing to AEM's start URL (where the port number is the one you defined on installation):
http://localhost:4502/
In case you already opened the browser, enter the URL manually.

In the appearing login dialog box, enter the default administrator's credentials (admin/admin), and then click **Sign In**.

AEM Login Dialog Box

The Welcome screen appears, displaying the different possibilities to continue. The next exercise, discusses the Websites console.



AEM Welcome Screen

Congratulations! You successfully installed and started AEM.

Command Line Start

First, you may want to know which parameters are available to configure quickstart prior to running the installation. Therefore, enter the following command to display a complete list of optional parameters:

```
java -jar cq-author-4502.jar -h
```

The AEM quickstart installer will show all available command-line options without starting the server.

In addition, you need to tune the JVM used for running AEM. Tuning the JVM is an important and delicate task and requires a more realistic environment in terms of resources (hardware, operating system, and so on) and workload (content, requests, and so on). For now, it will be enough to know that you can start your instance (Author or Publish) using the following parameters:

-Xms --> assigns the initial heap size

Default value	64 MB for a JVM running on 32-bit machines, or 83 MB for 64-bit machines
Recommended	Specific to physical memory available and expected traffic
Syntax	-Xms512m (sets the initial heap size to 512 MB)

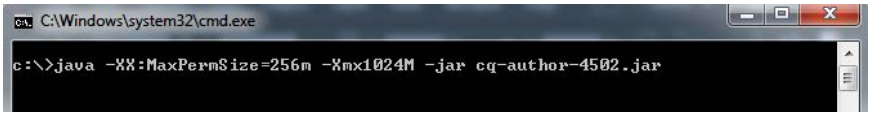
-Xmx --> assigns the maximum size the heap can grow

Default value	64 MB for a JVM running on 32-bit machines, or 83 MB for 64-bit machines
Recommended	Specific to physical memory available and expected traffic, but should be equal or greater than the initial size. To run AEM, it is recommended to allocate at least 1024 MB of heap size.
Syntax	-Xmx1024m (sets the maximum size the heap can, in the example we are letting it grow to 1024 MB; however, in production, this should be higher because AEM does consume a lot of resources on this aspect)

-XX:MaxPermSize --> assigns the heap to hold reflective data of the VM (for example, Java objects)

Default value	32 MB for a JVM running as a client, or 64 MB when running as a server
Recommended	The 'PermSize' should be set to at least 128 MB for 'normal-sized' Web apps or 256 MB for larger Web apps with significant Java activity
Syntax	-XX:MaxPermSize=128m (sets the initial perm gen size to 128MB)

You can now install/start AEM from the command line together with increasing the Java heap and perm gen size, which will improve performance. See the image below for an example of the command line.



AEM command line start

For this training, you can start the AEM installation process as follows:

- For a 32-bit Java VM enter the following:
  java -Xmx1024M -jar cq-author-4502.jar
- For a 64-bit VM, enter:
  java -XX:MaxPermSize=256m -Xmx1024M -jar cq-author-4502.jar

NOTE: You can also add the parameter -gui to the end of the command line to provide a graphical interface, which displays the current installation step/status and an on/off button to easily stop the application like described in the previous section.

You can control the way AEM is installed by defining properties via file name. The following file name patterns can be used for the AEM quickstart JAR file:

```
Quickstart filename options
-------------------------------------------------------------------------------

Usage:
 Rename the jar file, including one of the patterns shown below, to set the
corresponding option. Command-line options have priority on these filename
patterns.
-------------------------------------------------------------------------------

-NNNN
        Include -NNNN.jar or -pNNNN in the renamed jar filename to run on port
        NNNN, for example: quickstart-8085.jar
-nobrowser
        Include -nobrowser in the renamed jar filename to avoid opening the
        browser at startup, example: quickstart-nobrowser-8085.jar
-publish
        Include -publish in the renamed jar filename to run cq5 in "publish"
        mode, example: cq5-publish-7502.jar
-------------------------------------------------------------------------------

The license.properties file
-------------------------------------------------------------------------------
  The license.properties file stores licensing information, created from the
  licensing form displayed on first startup and stored in the folder from where
  Quickstart is run.
-------------------------------------------------------------------------------

Log files
```

Optionally, AEM can be installed in a third-party application server. The online documentation under http://dev.day.com/content/docs/en/cq/current/howto/install_application_server.html explains these steps in more detail.

For more in-depth information regarding the installation process, visit:
http://dev.day.com/docs/en/cq/current/getting_started/download_and_startworking.html

UI Introduction and Page Actions

Overview

This chapter gives an introduction to the AEM Touch-Optimized UI and page actions in general.

Objective

This chapter explains how to use the AEM Author Console to load and edit a page. This is important because you will use the website's administrator console to create and publish content throughout the course. In addition, you should understand the interfaces used by your author community.

The AEM UIs

The web-based authoring and administrative interfaces use AJAX to enable a desktop-like user experience. For example, when editing content on a website, authors can drag and drop elements like text paragraphs and images right onto the page and immediately see how their changes affect the appearance of the page.

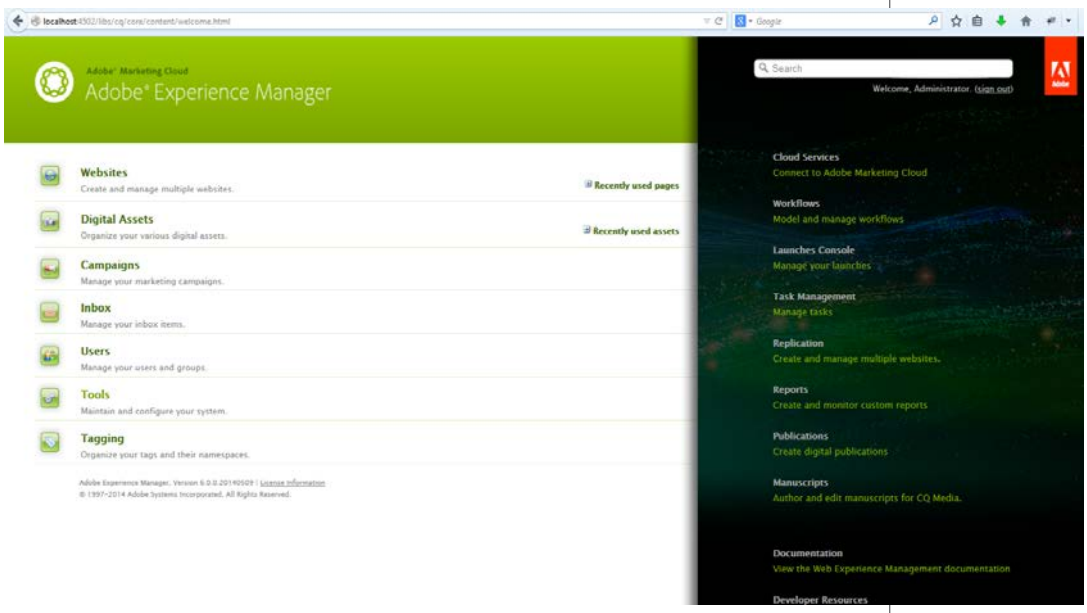
The AEM UI combines the advantages of a web interface with the fluidity and responsiveness usually associated with desktop applications. The functionality of AEM is made available through various specialized Web Consoles such as:

- **Sites** for creating and managing multiple websites
- **Assets** for organizing various digital assets
- **Social Communities** for moderating content of the social network
- **Forms** for adaptive, accessible forms for mobile and desktop
- **Inbox** for managing your inbox items
- **Security** for managing user accounts and groups
- **Tools** for maintaining and configuring the AEM system
- **Tagging** for organizing tags and their namespaces
- **Workflows** for modeling and managing workflows
- **Reports** for creating and monitoring custom usage reports
- **Packages** for installing, packaging, and sharing applications
- **Replication** for configuring Replication Agents
- **CRXDE Lite** for developing applications with a web-based IDE

The admin consoles for each major AEM function (Sites, Assets, Communities, and so on) present a touch-friendly console, which allow you to manage content using touch device paradigms. These Web Consoles are accessible from the Welcome page.

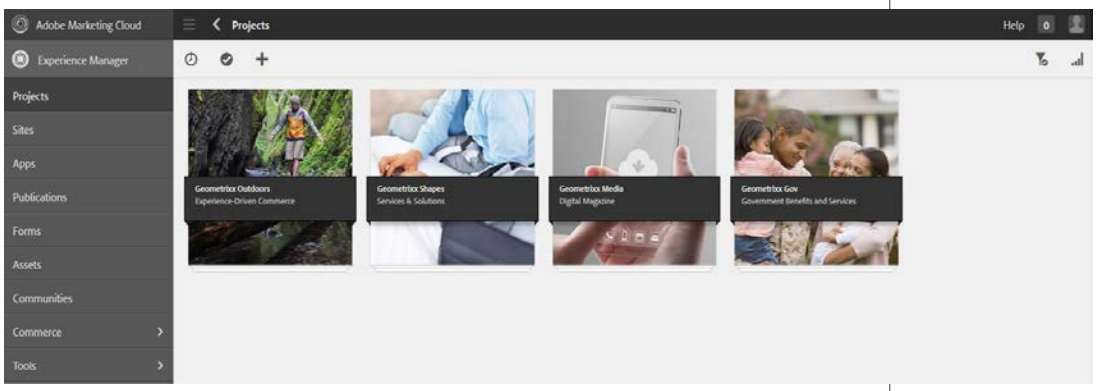
AEM Welcome Screen Utilities

There are two kind of Welcome screens—Classic and Touch-Optimized. Both screens have links for tools and utilities that manage the AEM application and the repository. You can access the classic welcome screen at http://localhost:4502/libs/cq/core/content/welcome.html.



AEM Classic Welcome Screen

You can use the Touch-Optimized UI Project Console as the welcome screen that gives you direct access to your projects and the ability to navigate to other consoles. You can access the Touch-Optimized UI Welcome screen at http://localhost:4502/projects.html.



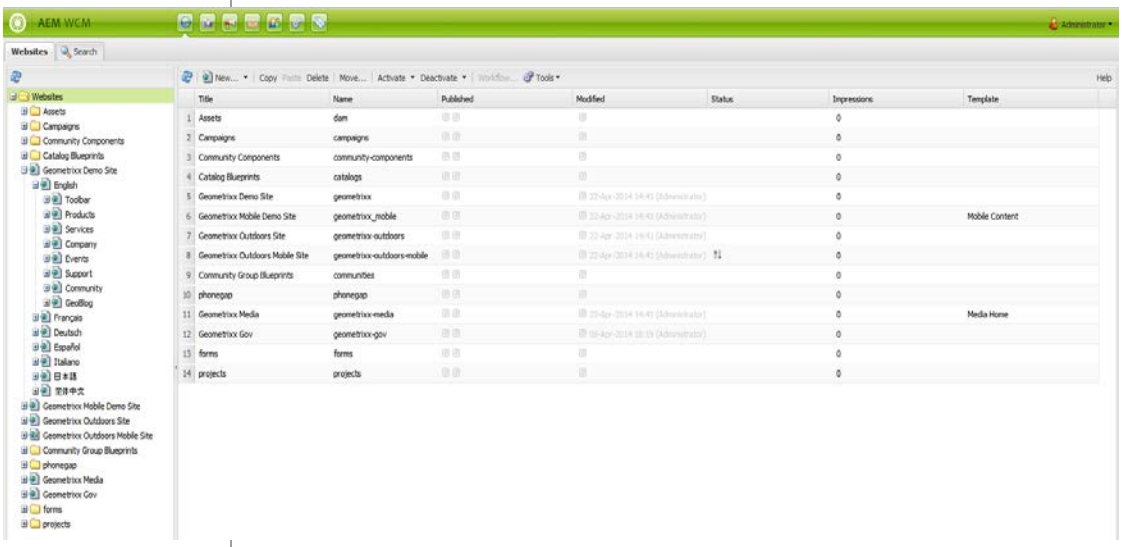
AEM Welcome Screen: Touch-Optimized UI

To go to a Web Console, click the appropriate icon. Once you are within a particular console, you can switch to another console using the Rail at the left navigation of the console page.

Websites Console (Classic Interface)

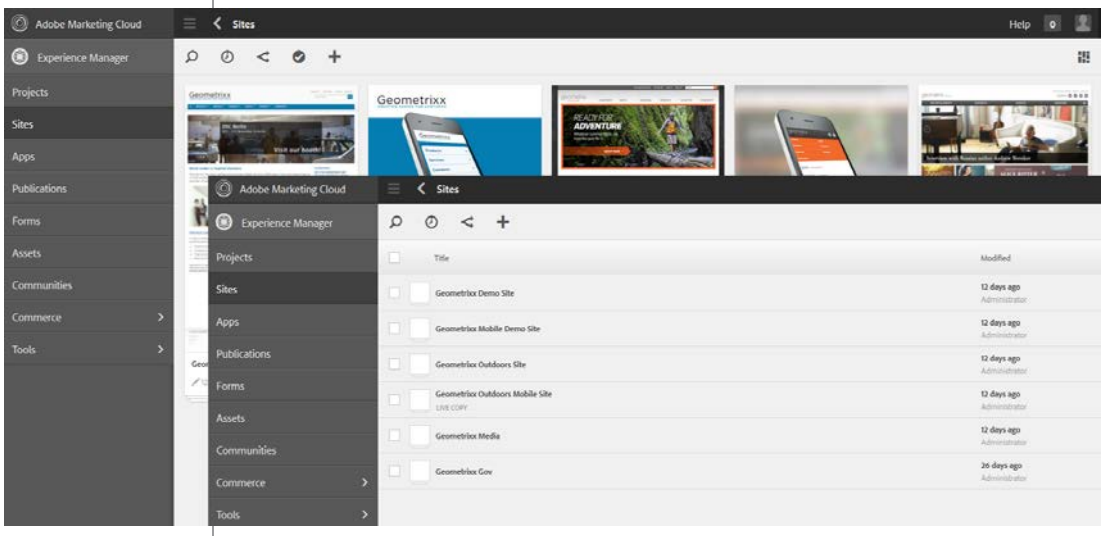
The Websites console lets you create, view, and manage websites running on your AEM instance. Through this console, you can create, copy, move, and delete website pages, start workflows, and activate (publish) pages. Double-clicking the page icon either in the explorer tree on the left or on the information grid on the right will take you directly to that page and let you edit it. This console is also referred to as the AEM WCM console.

With AEM 5.6, Adobe has introduced a new Touch-Optimized UI with responsive design for the Author instance. This differs considerably from the original Classic UI as it is designed to operate on both touch and desktop devices. All the new functions and features are being added only to the Touch-Optimized UI, so it is advisable that you always work on the Touch-Optimized UI.



Websites Console (Touch-Optimized UI)

In addition, the WCM Websites console presents a touch-friendly interface that authors and website administrators can access from mobile devices.

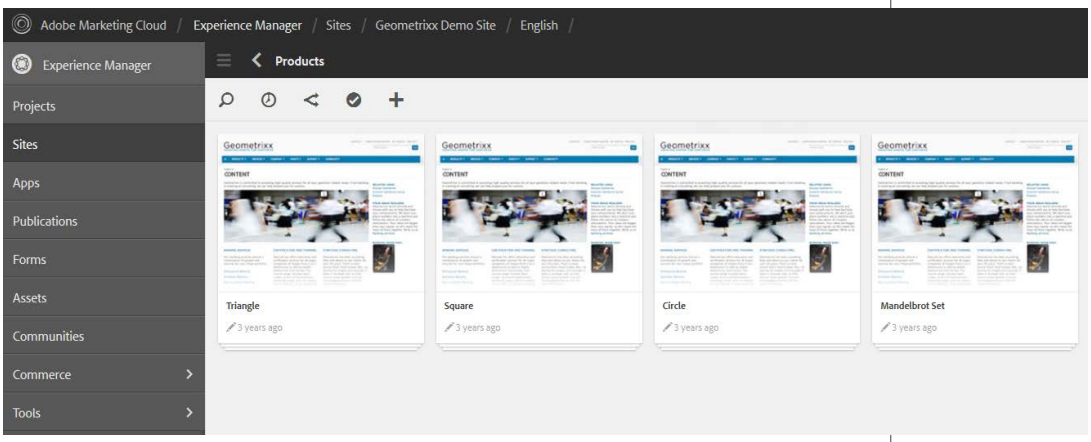


WCM Touch UI

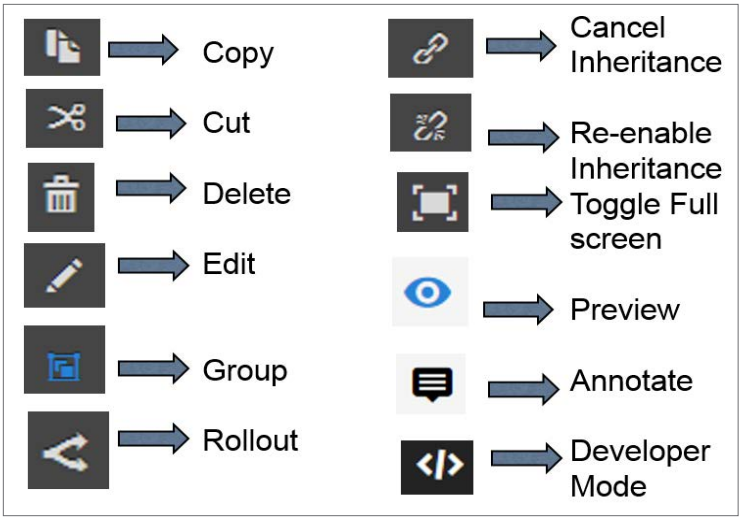
Navigating

You can navigate by:

- Clicking/tapping a specific item—for example, page or asset—to open the item (for navigating down the branch).
- Using the left navigation (Rail), select an item (for navigating up the branch, click on the appropriate section).



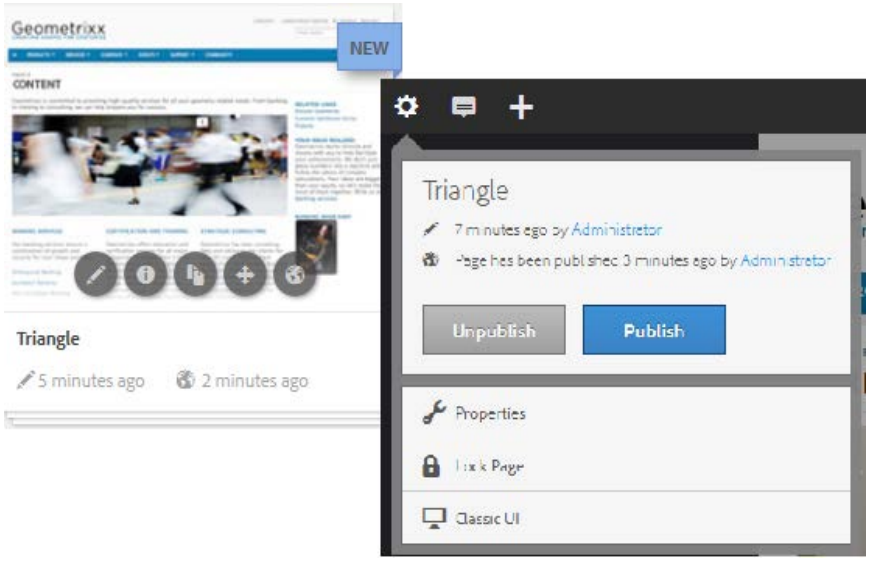
Navigation



Page Overview Information

When navigating through your console, items are represented as cards that provide an overview of relevant information. For example, for a page, the following information displays:

- A visual representation of the page content
- The page title
- When the page was last edited
- When the page was last published



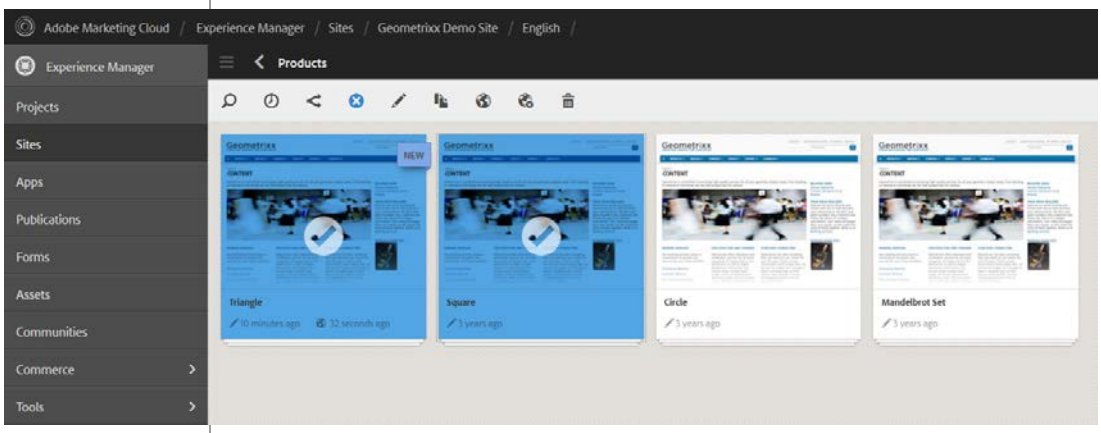
Selection Mode

Selection mode is a major concept of the Touch-Optimized UI. This determines whether clicking/tapping an item will:

- Navigate into the selected item, for example, navigate into sub-pages and child assets
- Select one or more items ready for action on those items, for example, selecting a page will allow you to then opt to view the page properties

The following two icons are used for entering and exiting selection mode:

- **In selection mode:**
  You can select one or more items. Additional icons display to reflect the actions you can take on the selected item.
- **Outside selection mode:**
  You can navigate through the items to your required location. You can also take action on one item using the quick actions available.



Quick Actions

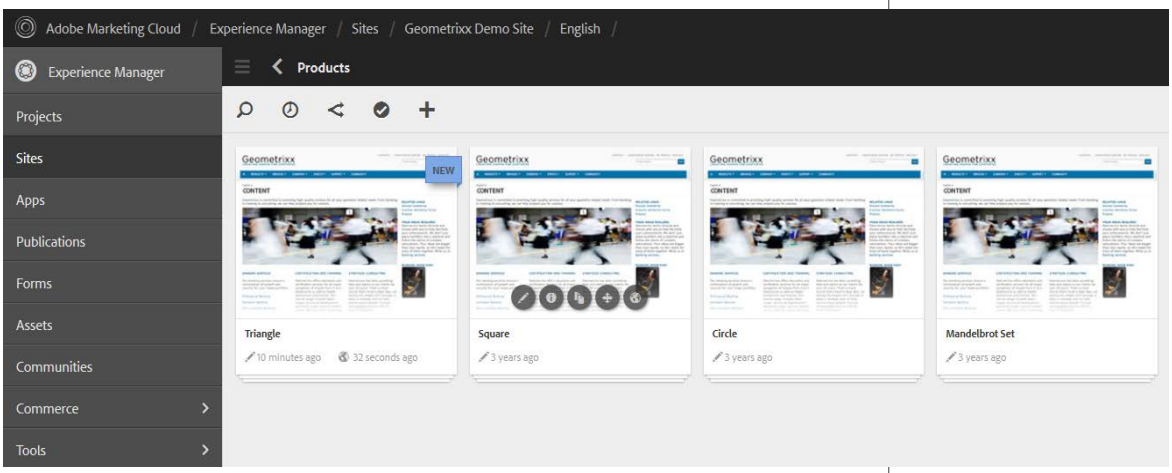
In addition, in selection mode, certain actions are available as quick action icons. Quick action icons are available for one item at a time.

The appearance of these is device dependent:

Device Type	Examples Include	Methods
Touch	Mobile device such as an iPad	Using touch-and-hold
Classic	Desktops and laptops	Using mouse-hover

Using touch-and-hold:

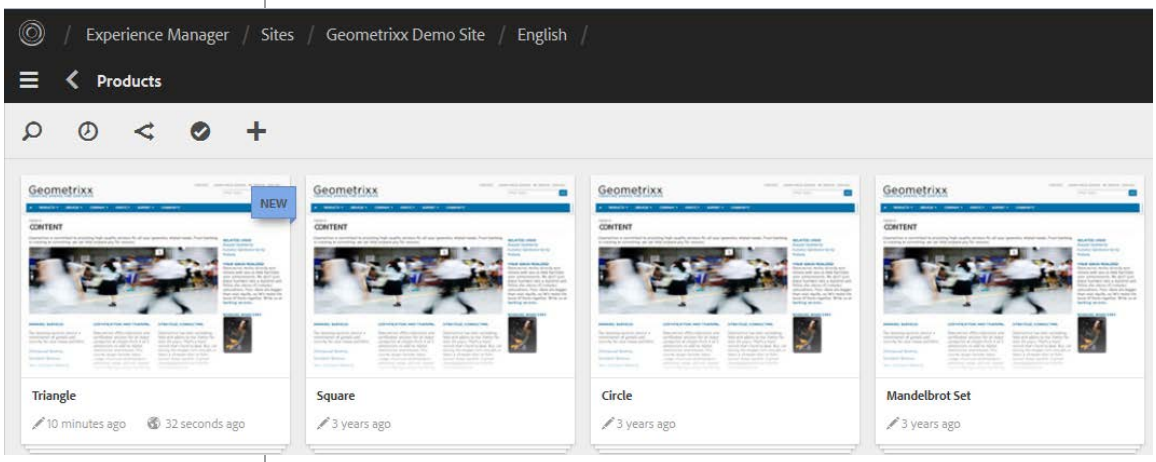
Using mouse-hover:



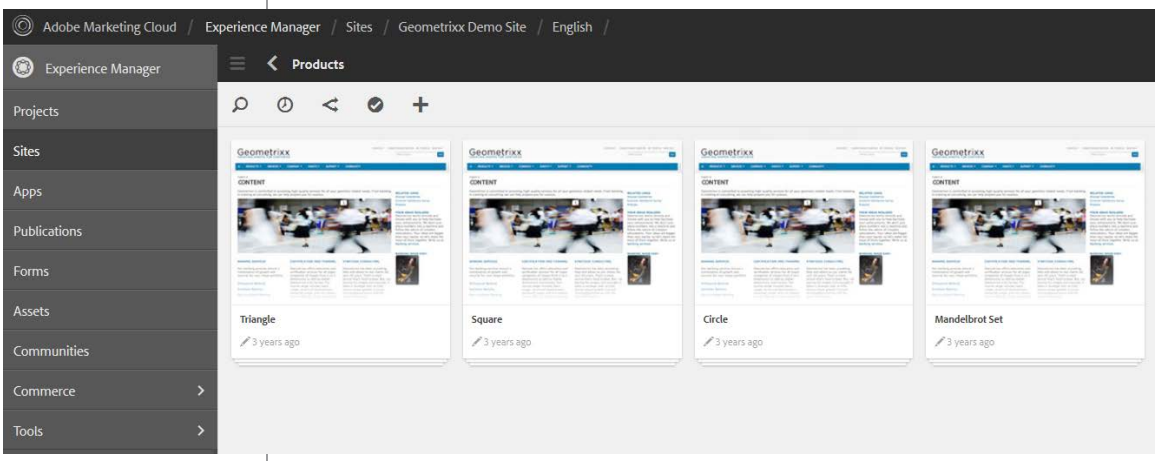
Rail - Toggle Open and Shut

Using the Rail icon, which acts as a toggle switch:

You can shut the left Rail:



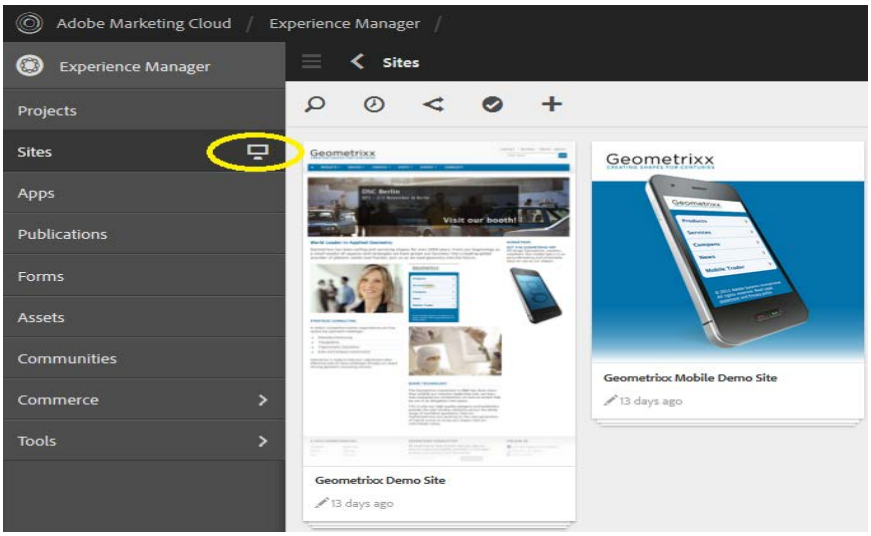
And open it again:



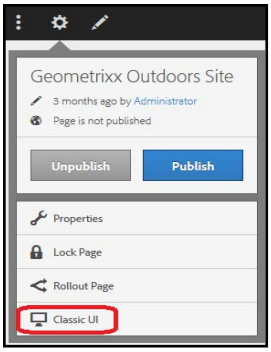
Accessing the Classic UI

The Touch-Optimized UI is now the default UI. Desktop users might want to revert to the Classic (desktop only) UI, for example, due to legacy or customization issues.

This is possible by clicking the icon that appears when your cursor is hovering over the currently active navigation item:



You can also change the Touch UI page to the Classic UI by clicking **Page Information -> Classic UI** when you are viewing a page.



Another way to open the Touch UI page in the Classic UI is that you replace 'editor.html' from the page URL to 'cf#'.

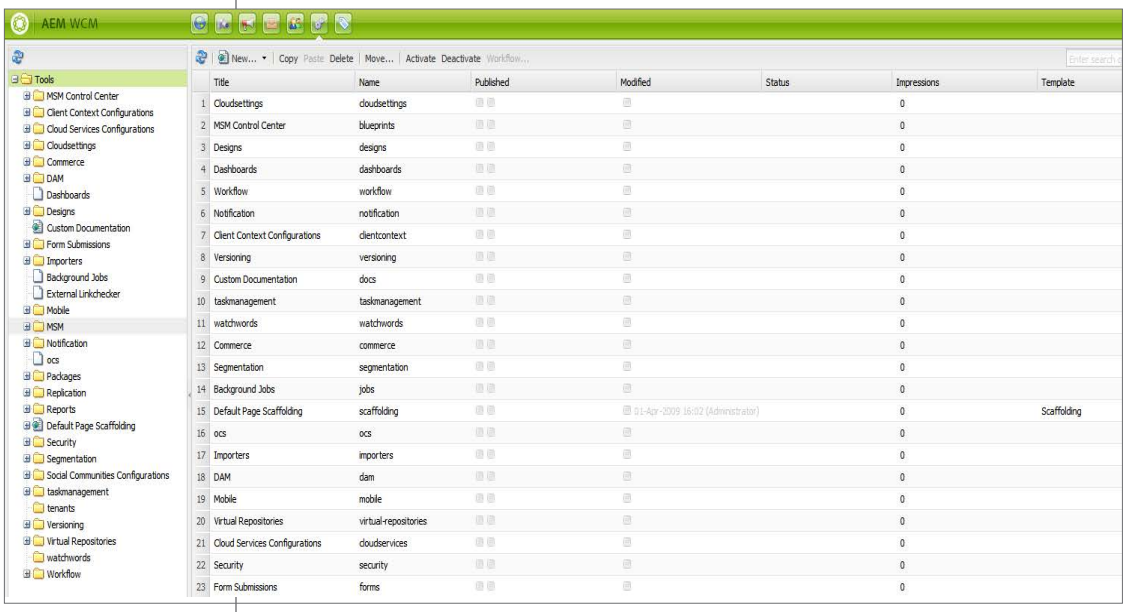
For example, the English page link for the Geometrixx Outdoor site in both UIs is as follows:

Touch UI: http://localhost:4502/editor.html/content/geometrixx-outdoors/en.html

Classic UI: http://localhost:4502/cf#/content/geometrixx-outdoors/en.html

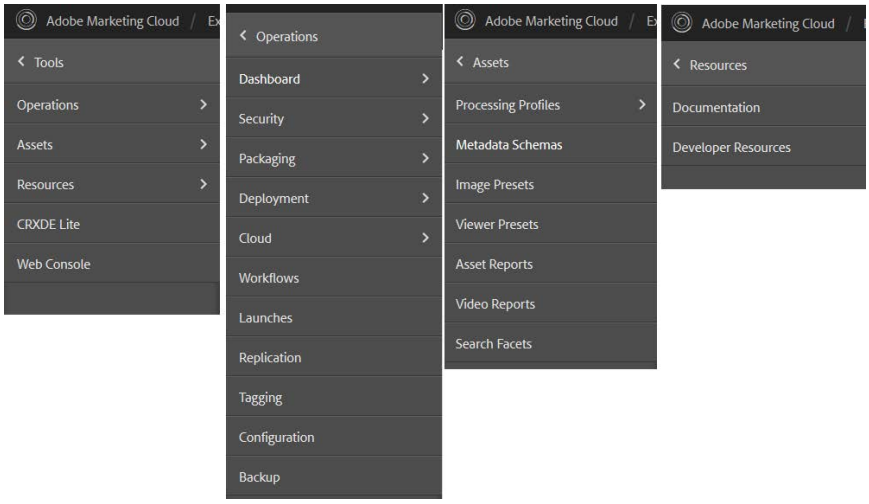
Tools Console

The Tools console provides access to a number of specialized tools that help you administer your websites, assets, and other aspects of your content repository.



Tools Classic Console

This provides a new interface to options available on the Tools console in the left navigation menu. You will experience the quick response while navigating and clicking the desired tool item. The Touch-Optimized UI Tools menu has all the new features and reports for your system administration task.



Tools Touch UI Left Navigation

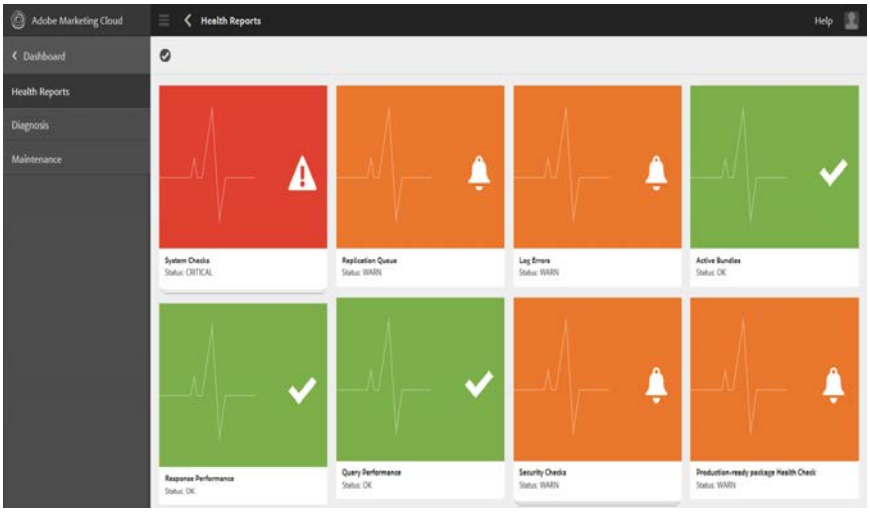
NOTE: Some of the Tools options above actually use the Classic UI.

Explore the Touch UI for Available Admin Tools

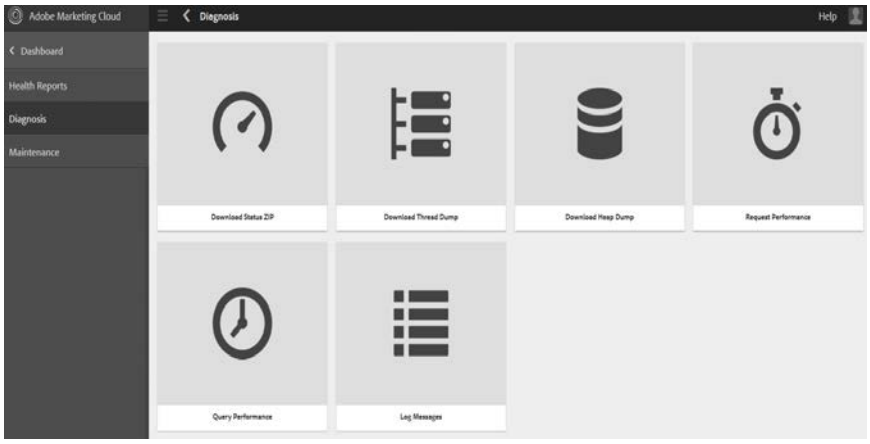
AEM 6.1 has two UIs and it is recommended to use the Touch UI, as it has all the new reports and functionalities. The improved navigation helps in reaching the desired tool in the Rail so that unnecessary page loads are avoided. When you click the desired

tool, you will be redirected to the corresponding page and it will be displayed in the right pane.

We have new health dashboards targeted for system admins. The dashboard shows the status of various parts of AEM and highlights any areas where problems exist. These dashboards can also show when maintenance tasks must be run.



Health Reports of Your Instance

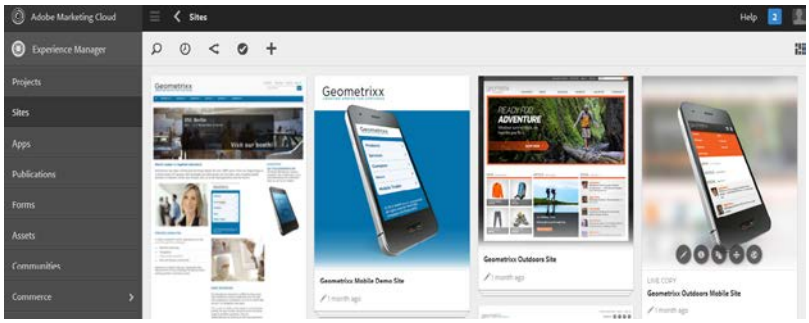


Diagnosis

This section is covered in detail later.

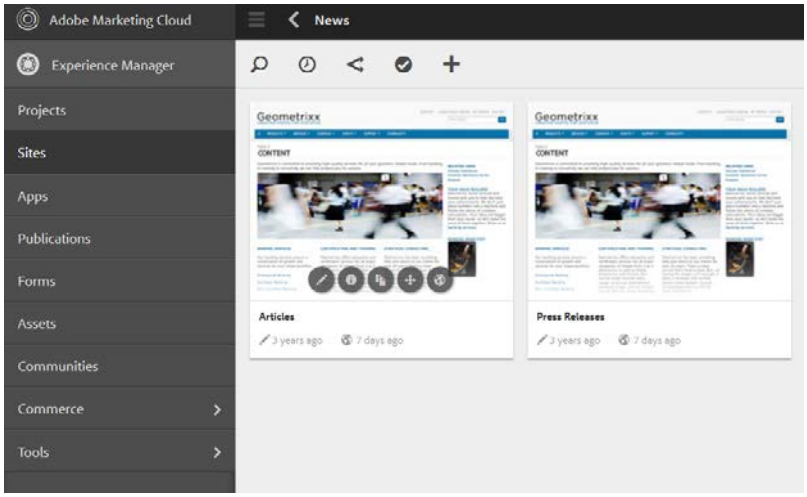
Exercise 2.1
Edit a Page in the Touch UI

1. Navigate to http://localhost:4502/projects.html. This will take you to the Project page, where authors manage their project-related tasks.



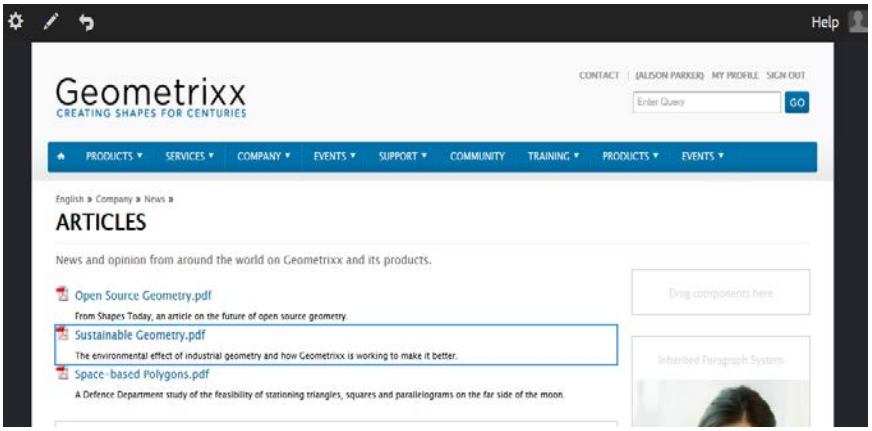
Websites Administrator Console

2. In the Rail (dark column at the left of the window) on the left pane, click **Sites**, and navigate to the page, **Geometrixx Demo Site** > **English** > **Company** > **News** by clicking the cards.



3. When the card for the **Articles** page is visible, move the cursor over it, and click on the **Open** or **Edit** icon(Pencil icon). You can open a page to be edited by one of following methods:
   a. From the Selection tool, select the page, and then click Open.
   b. You can directly hover the cursor over the page and click **Open**.

After you have opened a page, you can navigate to other pages within the site to edit them by clicking the links.

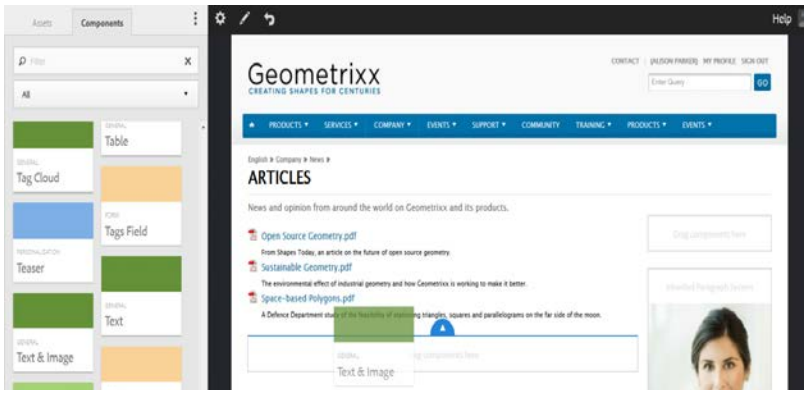


Page in Edit Mode

After you open the page, you can start to add content. You do this by adding new paragraphs or editing existing paragraphs (also known as components).

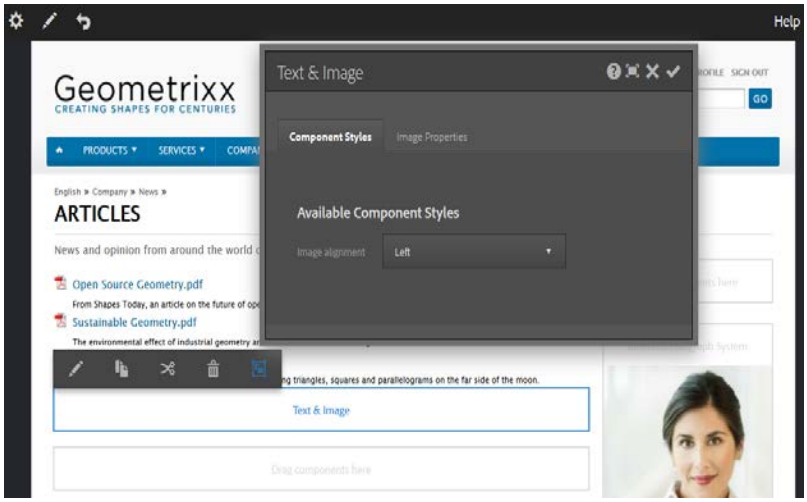
To insert a new paragraph, click the Toggle side panel, and then click the components. You can drag components or assets here to insert a new paragraph. This area appears wherever new content can be added, such as at the end of a list of other previously added paragraphs or at the end of a column.

4. Drag Te**xt & Image** from the component pane to the center of the dotted rectangle with the watermark text **Drag components or assets here**. The blue border on top of the area indicates that drag-and-drop is allowed.



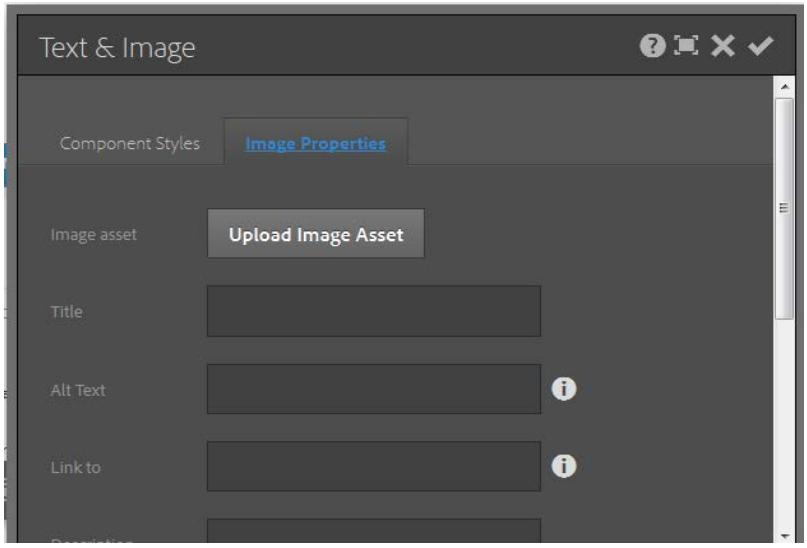
Webpage with the List of Available Components on the Component Tab

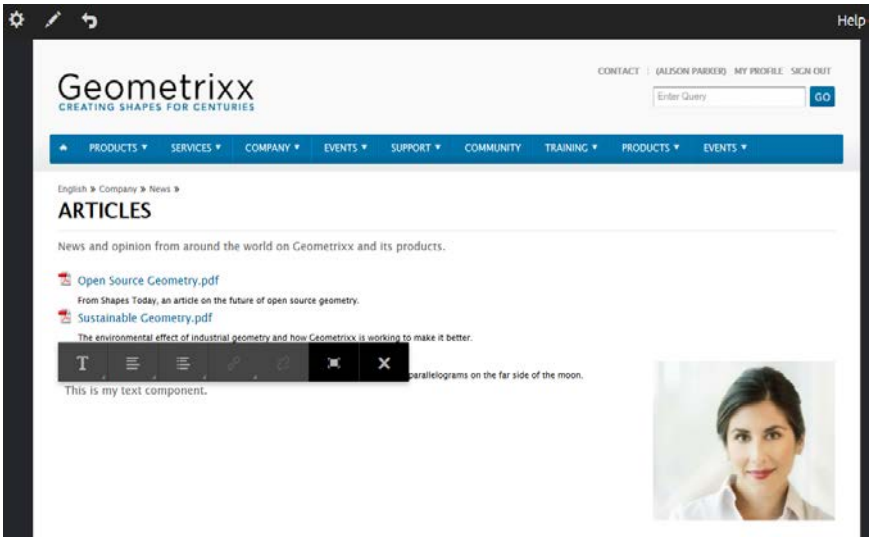
5. Double-click the thumbnail placeholder for the component to open the dialog box.



Dialog Box to Enter Content

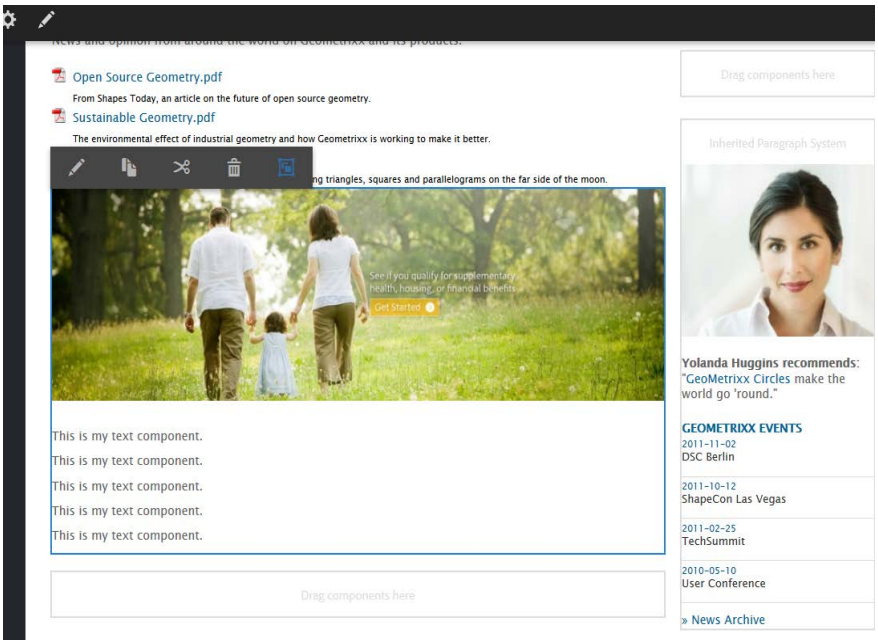
6. Click the **Image Properties** tab to open the **Image** pane of the dialog box. Select an image from hard drive or drag and drop an image from the **Content Finder** (on the left side) to the page.





Dialog Box Text With a WYSIWYG, Microsoft Word-Like Editor

7. Click **Done** to save the content.



Web Page With Fresh Content

8. Check out the result.

Congratulations! You successfully edited an AEM page.

Exploring the Administrative Interfaces

Overview

This chapter will be exploring the AEM administrative interface, which will help us to configure AEM for authors/developers.

Objectives

This section explains how to browse the application/server interfaces associated with an AEM installation. This will enable you to use their administrative/configuration capabilities.

What Interfaces Exist?

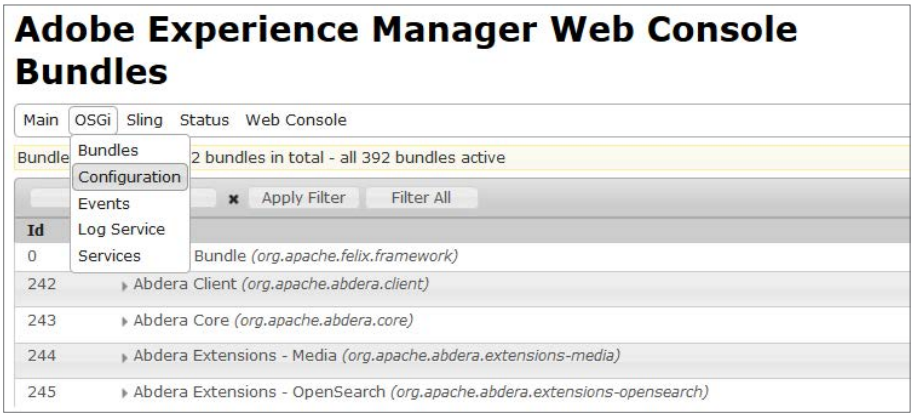
A typical AEM installation consists of a Java Content Repository (CRX) and a Launchpad (Apache Felix/Apache Sling) application. They each have their own web interface allowing you to perform the expected administrative and configuration tasks.

Exercise 3.1
Use the Adobe Web Console

1. Enter the URL http://localhost:4502/system/console in your web browser's address bar.
2. Enter the default administrator's credentials (**admin/admin**) in the dialog box, and then click **OK**. The **Adobe Experience Manager Web Console** appears, showing you the **Bundles** application.

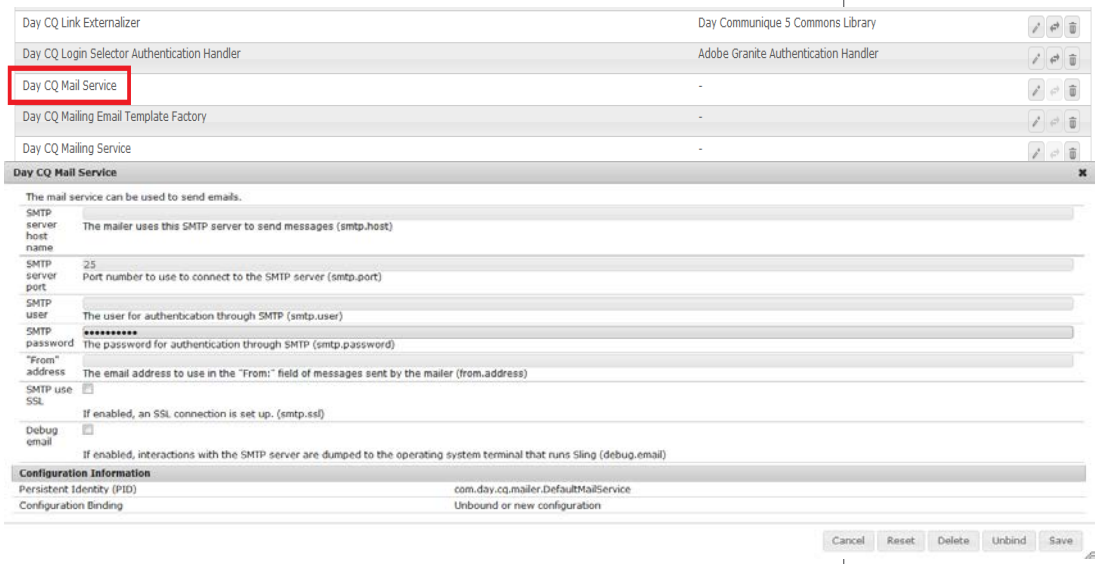
AEM Web Console

3. Click the **OSGi** drop-down menu, and select **Configuration**.



OSGi Configuration

4. Click on **Day CQ Mail Service**.

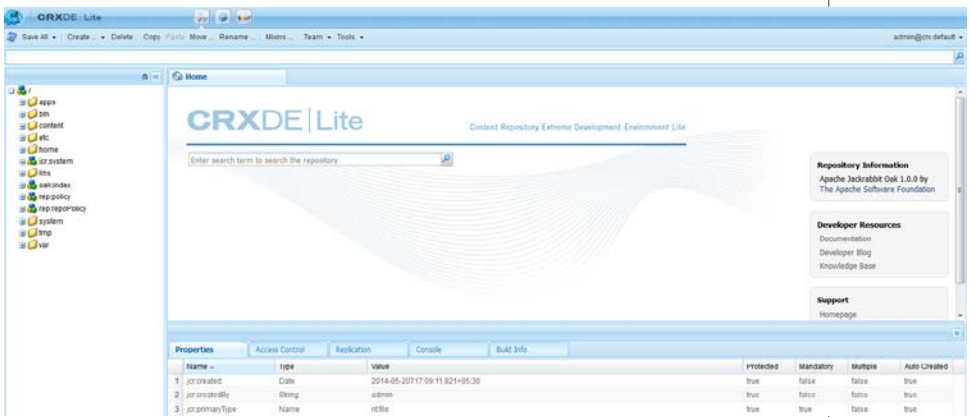


Congratulations! You successfully logged in to the AEM Web Console. From the Configuration Console, you can see which parts (bundles) of the AEM application can be configured.

Exercise 3.2
Use CRXDE Lite

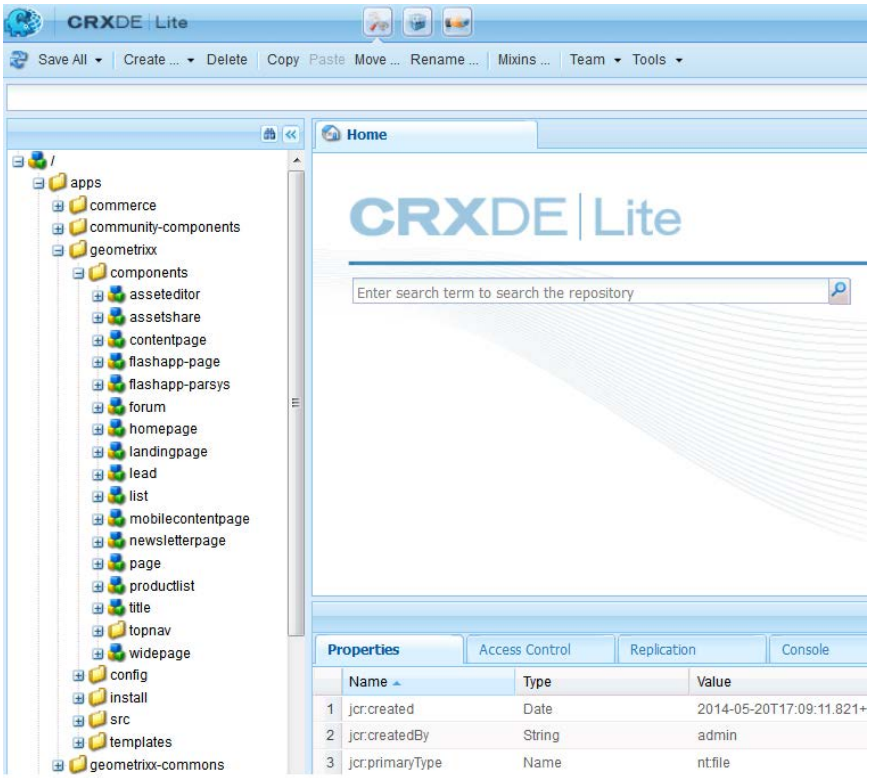
CRXDE Lite is a browser-based IDE, loosely based on Eclipse. The big difference between using CRXDE Lite and a common IDE like Eclipse or IntelliJ is that files created are stored directly through CRX. However, CRXDE Lite cannot check in files the into Subversion instance.

1. Enter the URL http://localhost:4502/crx/de/ in your favorite web browser's address bar or select the **CRXDE Lite** console from the **CRX Welcome** screen.
2. Make sure that you are logged in as a user with 'write' and 'execute' privileges, for example, like the admin user.



Web Console of the CRXDE Lite IDE

3. Navigate to the **/apps/geometrixx/components** folder to view the custom components created for the Geometrixx website/project.



Viewing the /components folder of the Geometrixx application

Congratulations! You successfully logged in to CRXDE Lite and browsed the custom components created for the Geometrixx website/project. Again, CRXDE Lite is embedded into AEM/CRX and enables you to perform standard development tasks in a web browser.

Chapter Four

Using the CRX Package Manager and Automating it With cURL

Objective

The following section explains how to create an AEM package that will combine all elements of the training project minus images. This is a good example of packaging application content, which you could then distribute to team members for review. To successfully complete and understand these instructions, you will need:

- The completed training project with appropriate extensions

Why Do You Need CRX Content Packages?

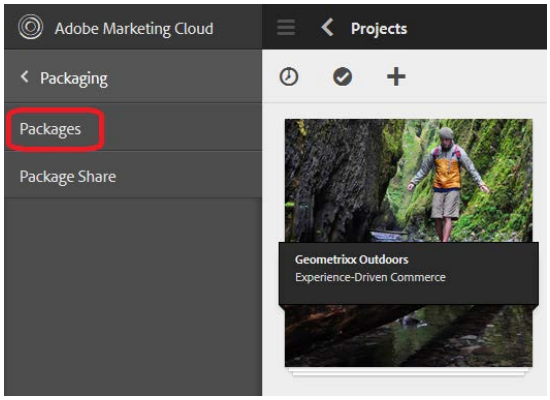
Packages can include content and project-related data. A package is a zip file that contains the content in the form of a file-system serialization (called filevault serialization) that represents the content from the repository as an easy-to-use-and-edit representation of files and folders. Content packages are a good way to, among other things, package application content, which you could then distribute to team members for review.

You can perform the following actions with packages:

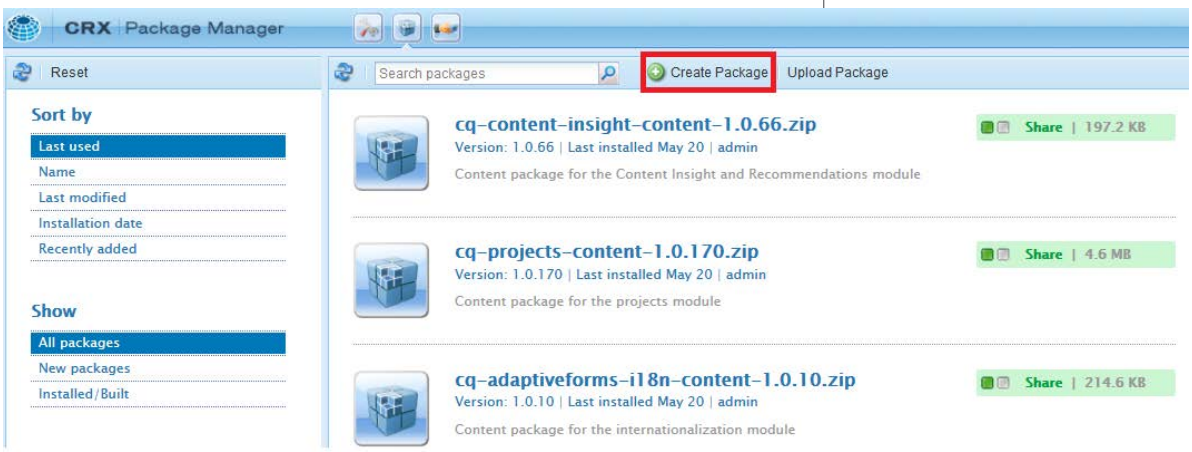
- Create new packages
- Modify existing packages
- Build packages
- Upload packages
- Install packages
- Download packages from the package share library
- Download packages from AEM to a local machine
- Apply package filters
- View package information

Exercise 4.1
Create, Build, and Download a CRX Package

1. Navigate to **Tools -> Operations -> Packaging -> Packages**. Select **Packages** in the Rail. You can directly open http://localhost:4502/crx/packmgr/index.jsp.

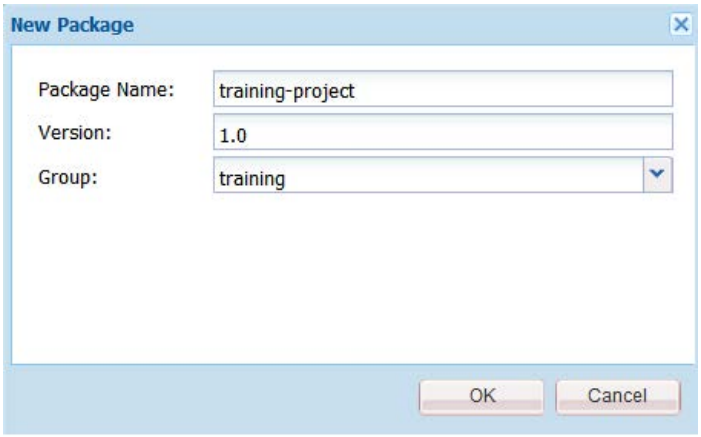


2. Click **Create Package** at the top.



AEM Packages: Create a New Package

3. Enter the **Package Name** (training-project), the **Version** of your package (1.0), and the package **Group** (training), and then click OK. You do not have to first create a folder for your package; you can enter the folder name (for example, training) in the **Group** field. You can also create a folder structure by entering the folder path, for example, training/exercises.

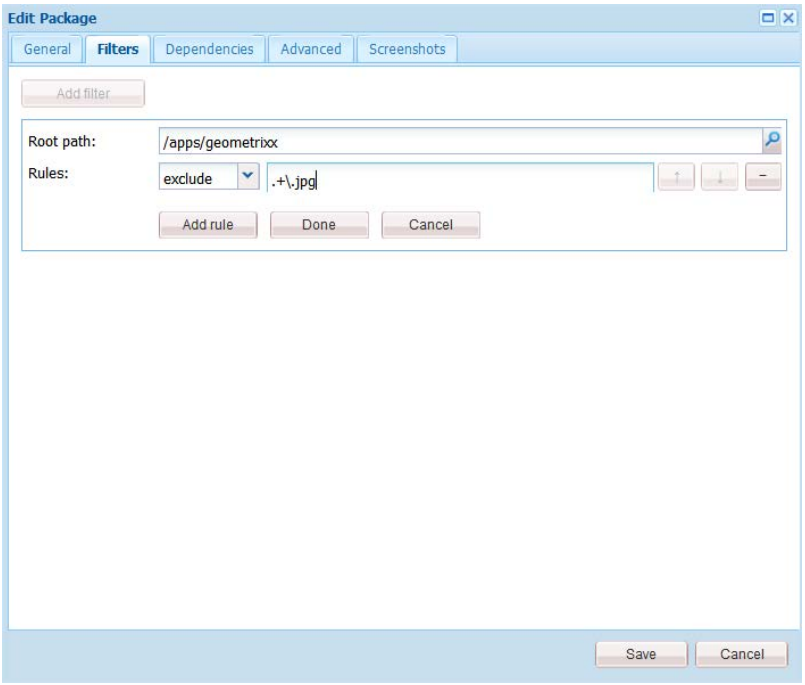


AEM New Package Dialog Box

4. Click **Edit** on the newly created package.
5. Select the **Filters** tab, and then click **Add filter definition**. Provide the following input for the fields:
   Root path: /apps/geometrixx
6. Click **Add rule**. Select **Exclude** from the drop-down list.

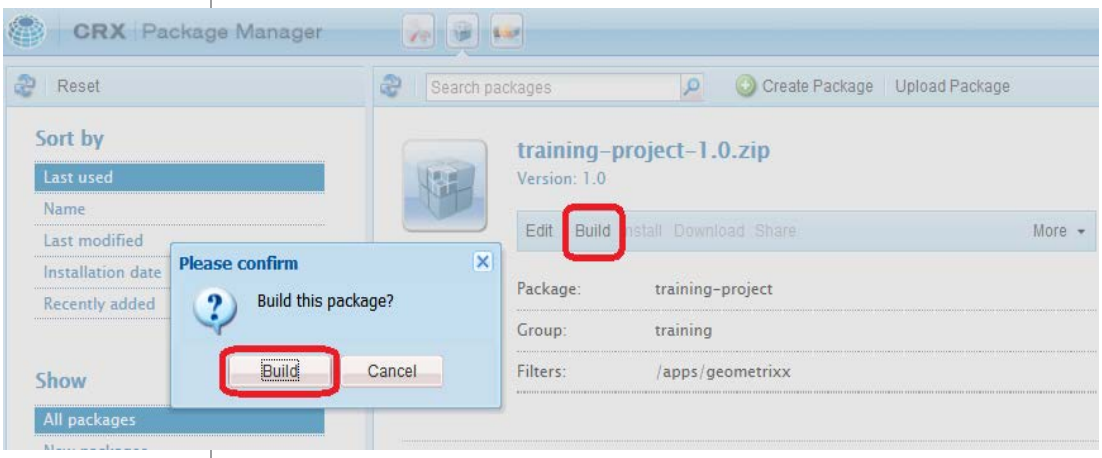
NOTE: The group name is a folder name where your package will be stored; you can choose an existing one or create a new one by providing a new name for the group.

7. Define a rule that excludes all jpg files (Exclude => .+\.jpg). Click **Done,** and then click **Save**.

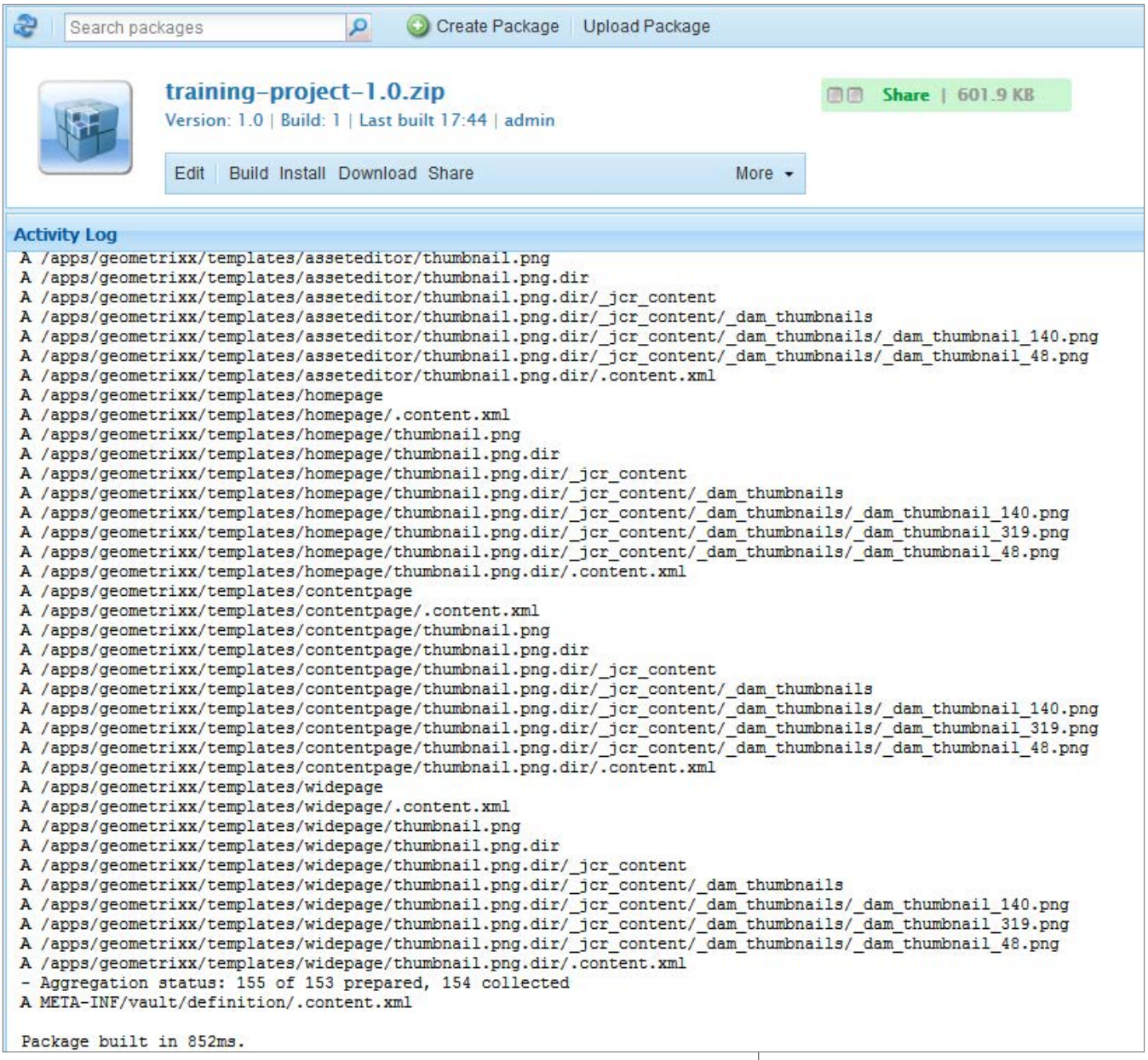


Filter Definition Dialog Box

8. Select **Build** to build the package.



Package Build Selection



Package Build Information

After the build process is successful, your package is ready for distribution and other users can install it to achieve the same output.

NOTE: A=Added Node, U=Updated Node, D=Deleted Node, and E=Error. Always check for errors in the build process. Although errors are seldom seen, verify and fix them immediately because they could cause instability with the repository at a later stage.

An HTTP service interface is available and allows for managing packages using command-line clients, like cURL or, wget or, automated scripts. In this way, you can automate content package management using scripts.
In this chapter, you will learn about the cURL syntax for content package management.
To successfully complete and understand these instructions, you will need:

- A running AEM author instance
- Optional: cURL HTTP client (if necessary, you can download cURL from: http://curl.haxx.se/download.html)

Configuring cURL for Windows

1. Download the cURL zip file from http://curl.haxx.se/download.html
   Extract the contents (if you have downloaded the correct version, you should find curl.exe)
2. Place **curl.exe** in a folder where you keep your software (for example, **D:\software\curl\curl.exe**).
3. To run cURL from the command line:
   a. Right-click **My Computer**.
   b. Select **Properties**.
   c. Click the **Advanced system settings** link.
   d. Select **Advanced tab** and click the **Environment Variables** button.
   e. Under **System variable**, select **Path**, and click the **Edit** button.
   f. Add a semicolon before the **curl.exe** folder path (for example, **;D:\software\curl**)
4. You can now run cURL from the command line by typing:
   curl www.google.com

Installing cURL in Mac OS X

1. Download cURL from http://curl.haxx.se/download.html.
2. Open a terminal and change the directory to the folder where the above mentioned file was downloaded.
3. Extract the compressed file using the command below (replace curl-7.30.0.tar in the command below as per your download):
   `$tar zxf curl-7.30.0.tar`
4. Change the directory to the extracted cURL directory:
   `$cd curl-7.30.0`
5. Run the make file as follows, and install cURL:
   `$ make && sudo make install`

When prompted for a password, enter the password for the super user account. When installed, a success message is displayed. You can now execute cURL from the terminal.

Exercise 4.2
Use Package Manager with cURL

The following Package Manager operations are currently supported from the command line:

- Printing the help screen
- Listing packages
- Removing packages
- Building packages
- Installing packages
- Uninstalling packages
- Downloading packages
- Uploading packages
- Creating packages
- Deleting packages
- Performing a dry run installation
- Previewing packages
- Listing package contents
- Rewrapping packages

To trigger the above operations, send requests using the command-line client to the CRX repository. The response is sent back in XML format.

1. To open the help screen, use the following command in a command-line terminal window (for example, Putty):
   `curl -u admin:admin` http://localhost:4502/crx/packmgr/service.jsp

You should receive the following response:

```
<crx version="1.0.0" user="admin" workspace="crx.default">
  <request>
  </request>
  <response>
    <data>
      +------------+----------------------------------------+
      |  Arguments | Comment                                |
      +------------+----------------------------------------+
      |  cmd=help  | print this help                        |
      +------------+----------------------------------------+
      |  cmd=ls    | print a list of all packages           |
      +------------+----------------------------------------+
      |  cmd=rm    | remove a  package                      |
      |  name      | package name                           |
      |  [group]   | group name (optional)                  |
      +------------+----------------------------------------+
      |  cmd=build | build a  package                       |
      |  name      | package name                           |
```

```
      |  [group]   | group name (optional)                  |
      +------------+----------------------------------------+
      |  cmd=inst  | install a package                      |
      |  name      | package name                           |
      |  [strict]  | true to fail on error                  |
      |  [group]   | group name (optional)                  |
      +------------+----------------------------------------+
      |  cmd=uninst| uninstall a package                    |
      |  name      | package name                           |
      |  [group]   | group name (optional)                  |
      +------------+----------------------------------------+
      |  GET       | download a package                     |
      |            | (content-disposition header contains   |
      |            | the correct filename)                  |
      |  [cmd=get] | optional                               |
      |  name      | package name                           |
      |  [group]   | group name (optional)                  |
      +------------+----------------------------------------+
      |  POST      | upload a new package                   |
      |  file      | package to upload                      |
      |  [name]    | optional name                          |
      |  [strict]  | true to fail on install error          |
      |  [install] | automatically install package if 'true' |
      +------------+----------------------------------------+
    </data>
    <status code="200">ok</status>
  </response>
</crx>
```

2. List the packages currently available on this AEM instance:

```
curl -u admin:admin http://localhost:4502/crx/packmgr/service.jsp?cmd=ls
```

You should receive a response similar to the following:

```
<crx version="1.0.0" user="admin" workspace="crx.default">
  <request>
    <param name="cmd" value="ls"/>
  </request>
  <response>
    <data>
      <packages>
        <package>
          <group>adobe/granite</group>
          <name>com.adobe.granite.backup.content</name>
          <version>0.0.12</version>
```

```
          <downloadName>com.adobe.granite.backup.content-0.0.12.zip</
downloadName>
          <size>34283</size>
          <created>Fri, 3 Feb 2012 11:46:10 -0600</created>
          <createdBy>Adobe Systems Incorporated</createdBy>
          <lastModified></lastModified>
          <lastModifiedBy>null</lastModifiedBy>
          <lastUnpacked>Fri, 9 Mar 2012 15:30:09 -0600</lastUnpacked>
          <lastUnpackedBy>admin</lastUnpackedBy>
        </package>
        <package>
          .
          .
          .
          .
        </package>
      </packages>
    </data>
    <status code="200">ok</status>
  </response>
</crx>
```

3. Upload a package: You will find a sample package in the memory stick in **/exercises/import_package/training_import.zip.**

   ```
   curl -u admin:admin -F file=@training_import.zip http://local-
   host:4502/crx/packmgr/service.jsp
   ```

You should receive a response similar to the following:

```
<crx version="1.0.0" user="admin" workspace="crx.default">
  <request>
    <param name="file" value="training_import.zip"/>
  </request>
  <response>
    <data>
      <package>
        <group></group>
        <name>training_import</name>
        <version>1.0</version>
        <downloadName>training_import-1.0.zip</downloadName>
        <size>288342</size>
        <created></created>
        <createdBy>null</createdBy>
        <lastModified>Thu, 26 Mar 2009 07:03:50 -0600</lastModified>
        <lastModifiedBy>admin</lastModifiedBy>
        <lastUnpacked></lastUnpacked>
        <lastUnpackedBy>null</lastUnpackedBy>
```

```
      </package>
    </data>
    <status code="200">ok</status>
  </response>
</crx>
```

4. Enter the following command to install the package you just uploaded.
   `curl -u admin:admin -F name=training_import` http://localhost:4502/crx/packmgr/service.jsp?cmd=inst

You should receive a response similar to the following:

```
<crx version="1.0.0" user="admin" workspace="crx.default">
  <request>
    <param name="cmd" value="inst"/>
    <param name="name" value="training_import"/>
  </request>
  <response>
    <data>
      <log>
Installing content...
Creating snapshot for package :training_import:1.0
A META-INF
A META-INF/vault
A META-INF/vault/config.xml
A META-INF/vault/filter.xml
A META-INF/vault/nodetypes.cnd
A META-INF/vault/properties.xml
A /.content.xml
A /content
A /content/.content.xml
A /content/dam

. . .

       A/content/dam/photos/people.jpg/jcr:content/renditions/cq5dam.
thumbnail.48.48.png/jcr:content/jcr:data
       A/content/dam/photos/people.jpg/jcr:content/renditions/cq5dam.
thumbnail.140.100.png
       A/content/dam/photos/people.jpg/jcr:content/renditions/cq5dam.
thumbnail.140.100.png/jcr:content
       A/content/dam/photos/people.jpg/jcr:content/renditions/cq5dam.
thumbnail.140.100.png/jcr:content/jcr:data
saving approx 71 nodes...
Package imported.
Package installed in 971ms.
      </log>
```

```
    </data>
    <status code="200">ok</status>
  </response>
</crx>
```

5. To remove a package, enter the following command:
   `curl -u admin:admin -F name=training_import` http://localhost:4502/crx/packmgr/service.jsp?cmd=rm

You should receive a response similar to the following:

```
<crx version="1.0.0" user="admin" workspace="crx.default">
  <request>
    <param name="cmd" value="rm"/>
    <param name="name" value="training_import"/>
  </request>
  <response>
    <status code="200">ok</status>
  </response>
</crx>
```

Congratulations! You successfully automated uploading, installing, and removing a content package. Refer to the documentation for more information about building, uninstalling, and other cURL commands supported by Package Manager.

Configuring OSGi Bundles, Log files, and Setting Runmodes

Overview

OSGi is a fundamental element in the technology stack of AEM. It is used to control the composite bundles of AEM and their configuration.

OSGi provides the standardized primitives that allow applications to be constructed from small, reusable, and collaborative components. These components can be composed into an application and deployed.

This allows easy management of bundles as they can be stopped, installed, and started individually. The interdependencies are handled automatically. Each OSGi component is contained in one of the various bundles.

At a basic level, this can just be a case of moving a CRX content package to a new environment, uploading it to the repository, and importing its contents. Alternatively, the package can be activated using the Tools section of the AEM Admin Console. However, some consideration should be given to managing this process, to ensure that your applications can be deployed efficiently and without any problems.

AEM is installed with default settings for all parameters, which allows it to run out-of-the box. However, you can configure AEM for your own specific requirements.
This can be done either:

- In the repository (CRX)
- In the Apache Felix Web Management Console
- In the file system
- In AEM WCM

We will be discussing the first two items in the list in the next sections.

A subset of configurations is available in the CRX repository. This ensures that copying or replicating repository contents re-creates identical configurations. You can also add your own configurations, dependent on run mode, to the repository. Apache Felix Web Console management is the standard location for configuring OSGI bundles. A few configuration files reside within the file system. Similarly, various aspects can be

configured within WCM itself—many using the tools console like Replication Agent, Dispatcher configuration, and so on.

Configuring AEM is straightforward, but you must be aware that certain changes can have a major impact on the application(s). For this reason, ensure you have the necessary experience and knowledge before you start to configure AEM; make only the changes that you understand, know, and are required. Any changes made through the OSGi console are immediately applied to the running system (no restart is required).

Resolution Order at Startup

The following order of precedence is used:

1. Repository nodes under /apps/*/config....either with type sling:OsgiConfig or property files (CHECK)
2. Repository nodes with type sling:OsgiConfig under /libs/*/config.... (ootb defns)
3. Any .config files from <cq-installation-dir>/crx-quickstart/launchpad/config/.... on the local file system

This means that a generic configuration in /libs can be masked by a project specific configuration in /apps.

Structure Within the Repository

Directory	What it contains
/apps	Application related; includes component definitions specific to your website. The components that you develop can be based on the out-of-the-box components available at /libs/foundation/components.
/content	Content created for your website.
/etc	Tools section and the details about them.
/home	User and group information.
/libs	Libraries and definitions that belong to the core of WEM. The subfolders in /libs represent the out-of-the-box WEM features as, for example, search or replication. The content in /libs should not be modified as it affects the way WEM works. Features specific to your website should be developed under /apps.
/tmp	Temporary working area.
/var	Files that change and are updated by the system, such as audit logs, statistics, and eventhandling. The subfolder /var/classes contain the Java servlets in source and compiled forms that have been generated from the component scripts.

Considerations for Deploying Applications to Other Environments

- Make sure your application code is separate from the configuration, as you may need to use different configurations with the same application for different environments. Creating separate packages makes it easy to deploy the appropriate configuration for each environment.
- Make sure your application code does not depend on specific content, as this may not be present in all environments.
- Code packages will be created in the development environment and will then travel from development to test and then to production. Content will generally travel from production, to test, and then to development to be used as test content, so separate packages and processes may be needed.

Two approaches can be taken when creating packages for deployment so that the configuration information can be configured separately from the application code.

Packaging Style 1

Deploy two packages:

- One with the application code
- One with the environment-specific configuration for development/test/production

Packaging Style 2

Deploy one package with the configuration for all relevant environments together with the application code. With this packaging option, you use different run modes to select the correct configuration for the desired environment.

Version Manager

In addition to explicit purging by means of the purge tool, the Version Manager can be configured to purge old versions when new versions are created.

To configure the Version Manager, create a configuration for:
`PID com.day.cq.wcm.core.impl.VersionManagerImpl`

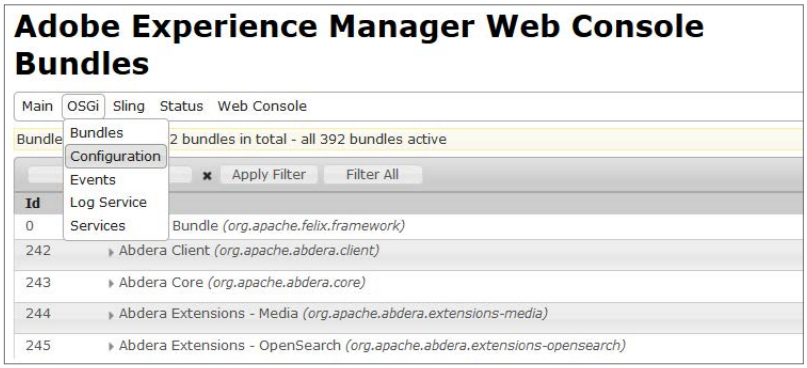
The following options are available:

`versionmanager.createVersionOnActivation` (Boolean, default: true)	Whether to create a version when pages are activated. A version is created unless the Replication Agent is configured to suppress creation of versions, which is honored by the Version Manager A version is only created if the activation happens on a path that is contained in the Version Manager.
`versionmanager.ivPaths` (String[], default: {"/"})	Paths on which versions are implicitly created on activation if versionmanager.createVersionOnActivation is true.
`versionmanager.purgingEnabled` (Boolean, default: false)	Whether to enable purging when new versions are created.
`versionmanager.purgePaths` (String[], default: {"/content"})	On which paths to purge versions when new versions are created.
`versionmanager.maxAgeDays` (int, default: 30)	On purge, any version older than this value will be removed. If this value is less than 1, purging is not performed based on the age of the version.
`versionmanager.maxNumberVersions` (int, default 5)	On purge, any version older than the n-th newest version will be removed. If this value is less than 1, purging is not performed based on the number of versions.

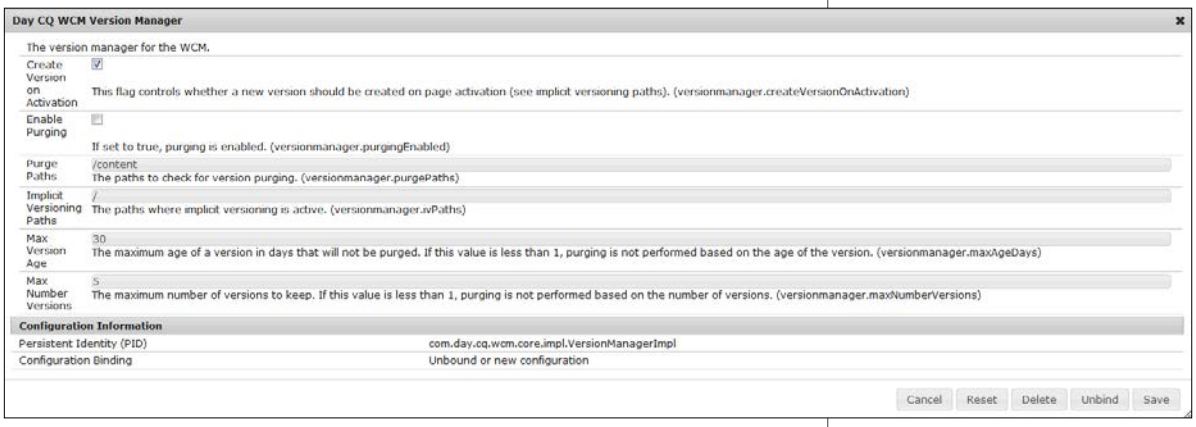
NOTE: The Version Manager can be configured with the Day CQ WCM Version Manager OSGi service.

Exercise 5.1
Configure the Version Manager

1. Navigate to the AEM Web Console: http://localhost:4502/system/console
2. Click the **OSGi** drop-down menu and select **Configuration**.



3. Select the **Day CQ WCM Version Manager**.



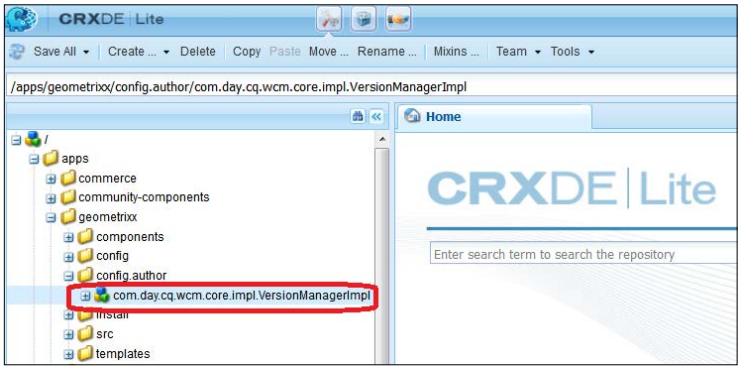
Although Adobe does not recommend that you use this console to configure bundles, you can find the information you need to enter the configuration information into the repository.
Notice the Persistent Identity (PID):
`com.day.cq.wcm.core.impl.VersionManagerImpl`

This is the name of the node you create in the repository.

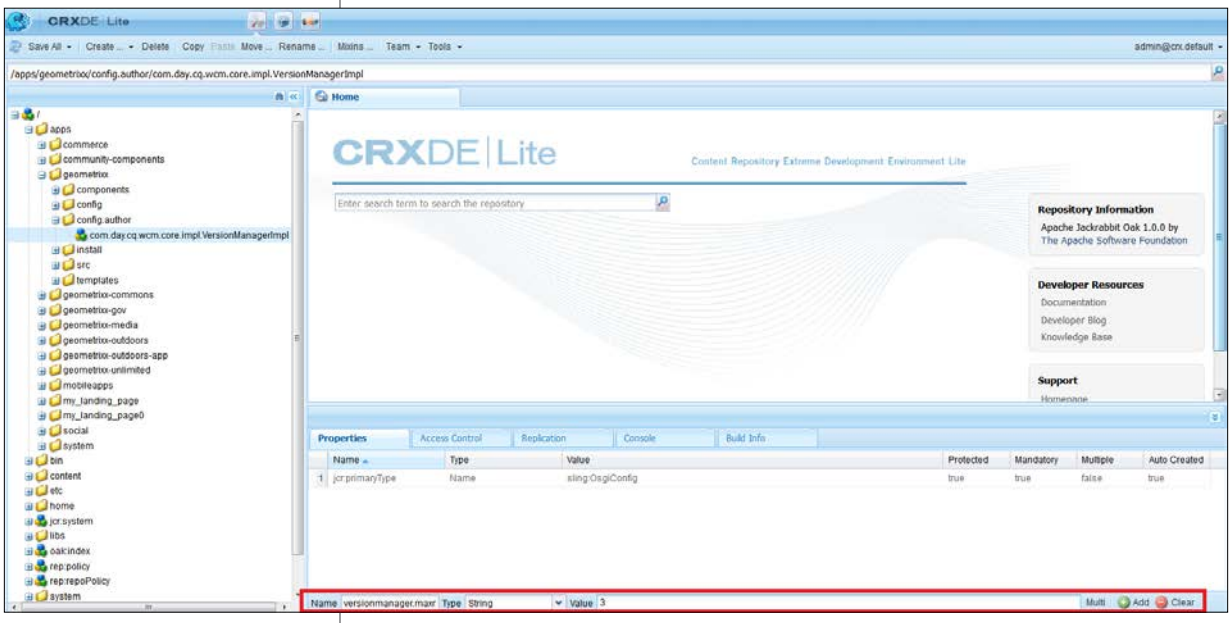
Notice the configurable items; for example, Enable Purging (versionmanager.purgingEnabled). The items in parentheses are the configurable property names.

4. Open CRXDE Lite: http://localhost:4502/crx/de/
5. Right-click **/apps/geometrixx**.
6. Click **Create -> Create Node.** Create a node of type **sling:Folder**, named **config.author**. Click **Save All**.

7. Right-click the new folder you just created, and click **Create -> Create Node.** Create a node of type **sling:OsgiConfig**, named **com.day.cq.wcm.core.impl.VersionManagerImpl**—the PID of the Version Manager, and click **Save**.



Creating properties in the repository using CRXDE Lite is done by adding the property information in the input fields at the bottom of the screen, and then clicking **Add**. Do not forget to click **Save**.



8. Add the following properties to the node, you just created:
   a. Create Version on Activation
      i. Name: versionmanager.createVersionOnActivation
      ii. Type: **Boolean**
      iii. Value:**true**
   b. Enable Purging
      i. Name:**versionmanager.purgingEnabled**
      ii. Type: **Boolean**
      iii. Value: **true**
   c. Purge Paths
      i. Name: **versionmanager.purgePaths**

ii. Type: **String**

iii. Value: **/content**

d. Implicit Versioning paths

i. Name: **versionmanager.ivPaths**

ii. Type: **String**

iii. Value: **/**

e. Max Version Age

i. Name: **versionmanager.maxAgeDays**

ii. Type: **Long**

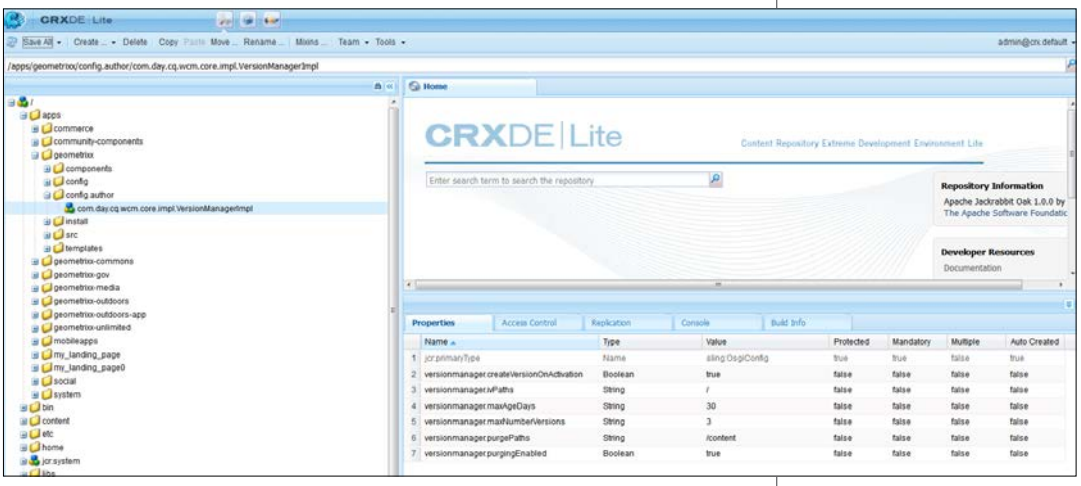
iii. Value: **30**

f. Max Number Versions

i. Name: **versionmanager.maxNumberVersions**

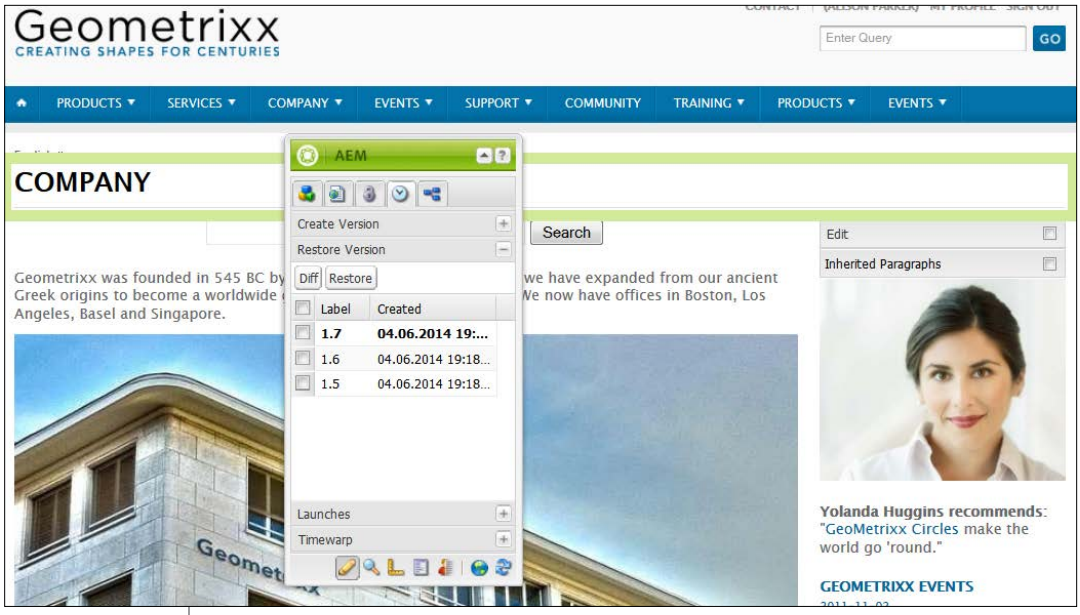
ii. Type: **Long**

iii. Value: **3**



9. Click **Save**.

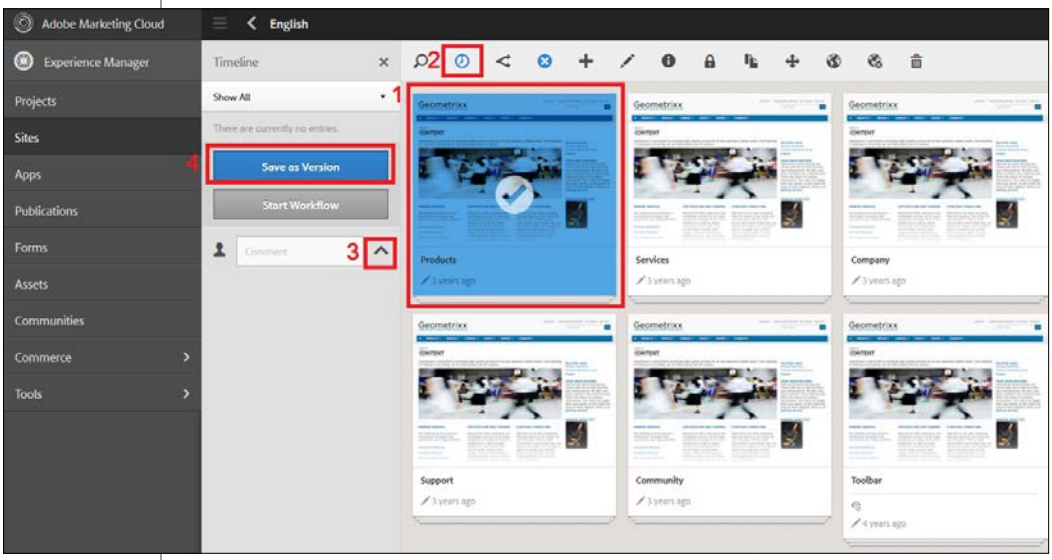
10. Test your configuration by opening a page in the Geometrixx website and saving a number of versions. You will notice that only the last three are versions available to restore.
That means the configuration is working as expected.



Exercise 5.2
Create a New Version Using the Touch-Optimized UI

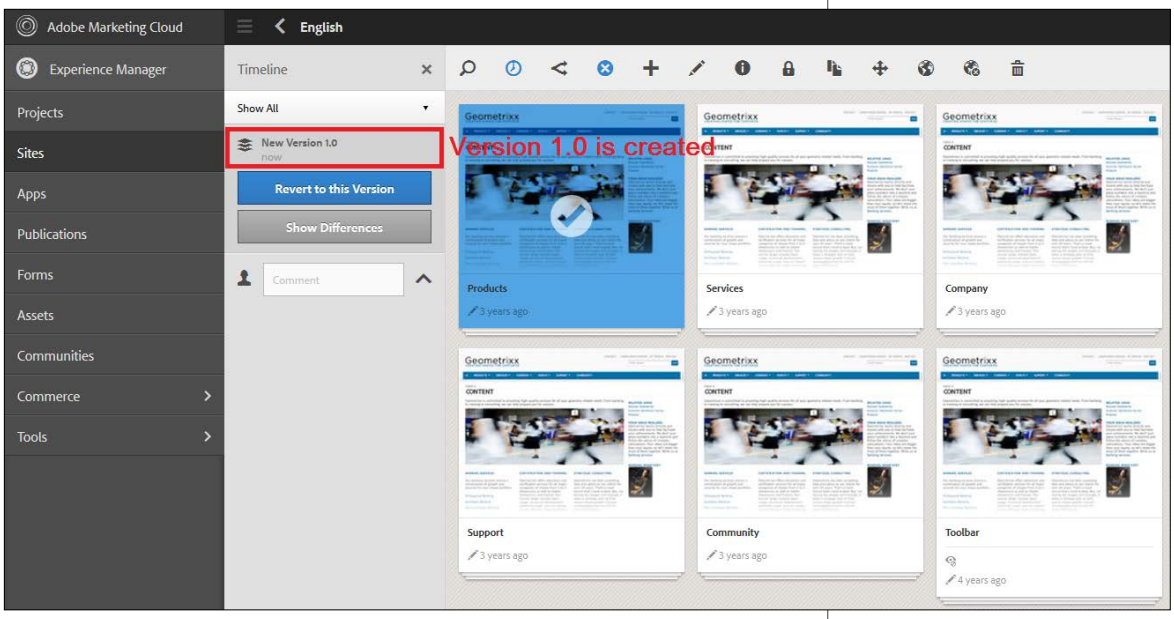
You can create a version of your resource from the timeline tab. Navigate to the page for which you want to create a version.

1. Select the page using selection mode.
2. Click **Timeline** (watch icon).
3. Click the ^ icon, which is next to the comment option.
4. Click **Save as Version**.



5. Click **Create**.

6. The information in the timeline will be updated to indicate the new version.



Replication and Reverse Replication Agents

Overview

Replication Agents are central to AEM as the mechanism is used to:

1. Publish (activate) content from an author to a Publish instance.
2. Explicitly flush content from the Dispatcher cache.
3. Return user input (for example, form input) from the Publish instance to the Author instance (under control of the Author instance).

Requests are queued to the appropriate agent for processing.

NOTE: Technically, a replication means copying content from one repository to the other. However, it is not like copying files. It is the typical repository approach with editing nodes and letting persistence managers store the content. A replication has a user assigned (an Administrator to define the access right), who replicates content to a user (author) on the receiving instance.

Set Up the Replication Agent for Two Publish Instances

Goal

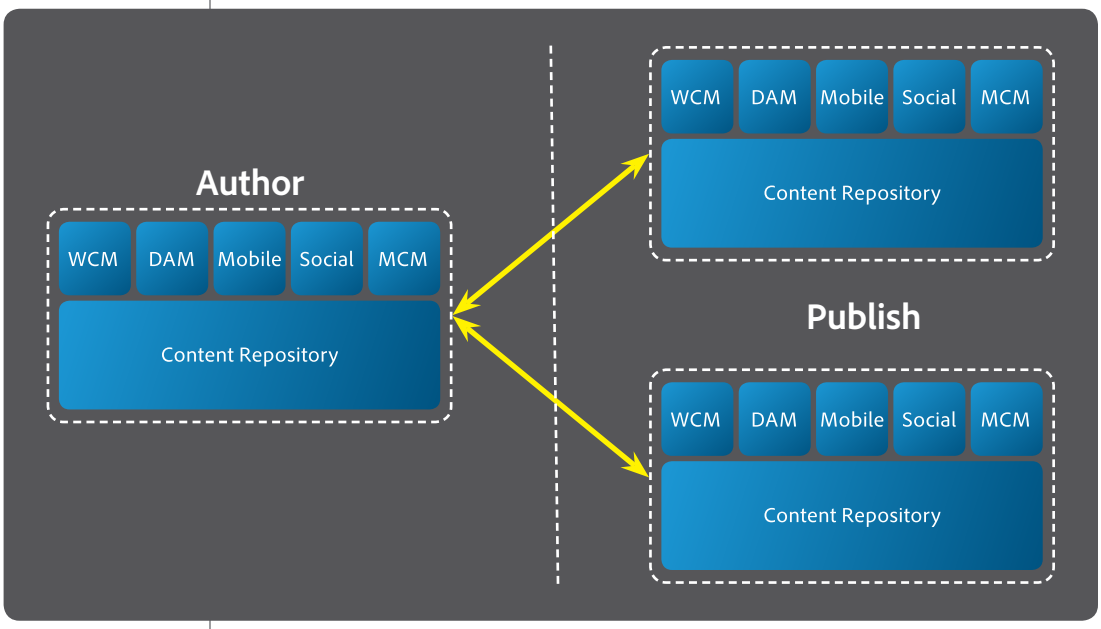
Configure Replication Agents so that they replicate/activate content to Publish Instances. Content replication allows administrators to specify subsets of content, usually in author instances, which will be automatically replicated to other repositories, mainly Publish Instances, elsewhere in a distributed environment. Replication can also take place when content entered by site visitors in Publish Instances has to be replicated into author instances.

A Replication Agent is a distinct point-to-point connection between a start point (current instance) and a specified target instance (usually a Publish Instance). You will need to specify in your environment as many Replication Agents as target instances you want to address. You need to create an agent because content replication has a clearly defined goal (as opposed to discrete tasks) and requires no continuous direct supervision or control.

The following instructions explain how to create and configure Replication Agents. To successfully complete and understand these instructions, you will need:

- A running AEM Author instance

AEM Environment Basic Setup



Replication Agents

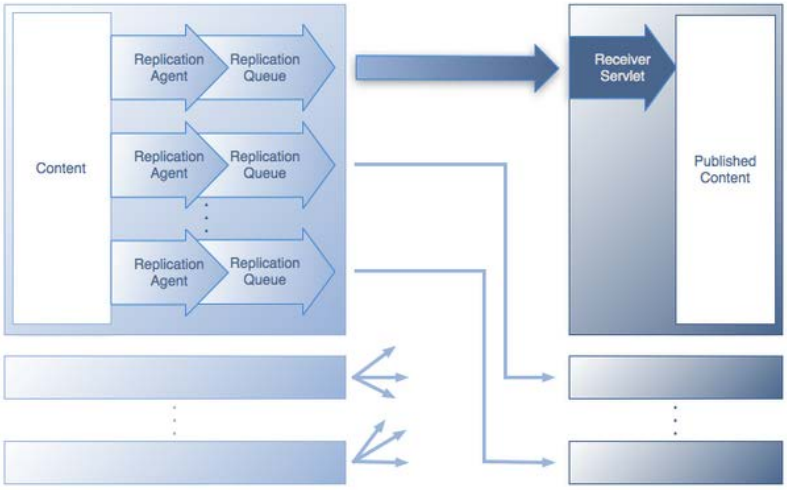


Diagram to Illustrate the Activation of Content from Author to Publish Instances

Replication to a Publish Instance takes place in several steps:

- The author requires that certain content be published (activated). This can be initiated by a manual request (when an author user checks the **Activate** button for a page in the AEM SiteAdmin console), or can be initiated automatically by triggers that have been previously pre configured as instructed in this exercise.
- The request is passed to any appropriate Replication Agent. An environment can have several Replication Agents. All available and enabled Replication Agents will be selected for replication requests.

- The Replication Agent (in the author instance) 'packages' the content and moves it to a replication queue for further processing (the actual replication).
- After a successful replication, the colored status indicator next to the web pages in the SiteAdmin console (Websites tab) is set to green (orange = in process, red = deactivated). This way, the author sees at any given time which pages have been replicated.
- The replication queue is processed and the content is copied to the 'other' instance (typically the Publish Instance) using one of the configurable protocols. Typically, the replication happens using the HTTP protocol.
- A servlet running on the receiving (usually publish) instance receives the request and copies the received content to the exact same location as indicated in the content package.
- The receiving servlet creates a new version of the existing content (can be disabled), and then replaces the content elements with the received ones.

Settings Tab Configuration Parameters

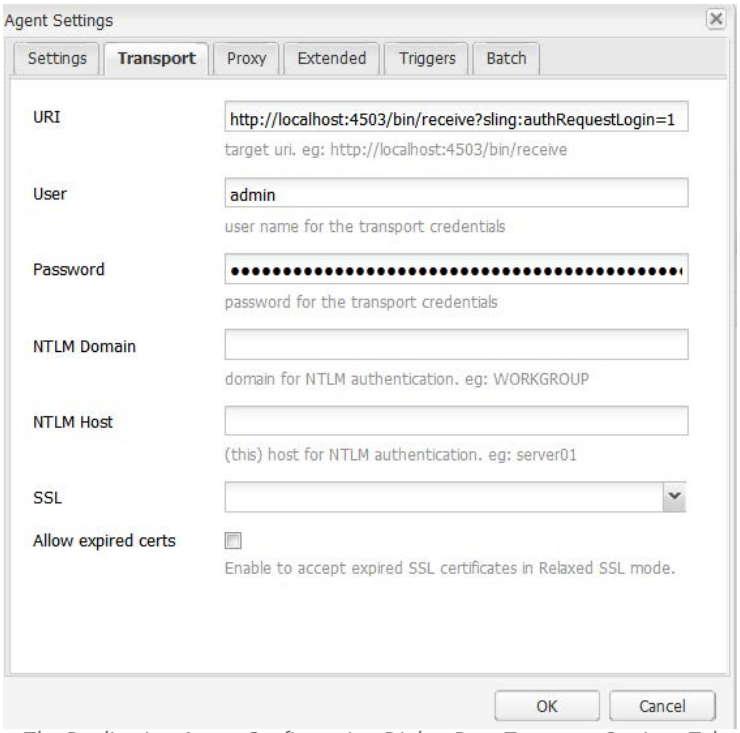
- **Name:** A unique name for the Replication Agent
- **Description:** A description of the purpose this Replication Agent will serve
- **Enabled:** Indicates whether the Replication Agent is currently enabled or disabled
- **Serialization Type:** The type of serialization:
  - **Default:** Set if the agent is to be automatically selected
  - **Dispatcher Flush:** Selected if the agent is to be used for flushing the web server (Dispatcher managed) cache
  - **Retry Delay:** The delay (waiting time in milliseconds) between two retries should a problem be encountered
  - **Default:** 60000
- **Agent User Id**: The agent will use this user account to collect and package the content from the Author instance. Leave this field empty to use the system user account (the account defined in Apache Sling as the administrator user; by default, this is 'admin'**)**.
- **Log Level:** Specifies the level of detail to be used for log messages:
  - **Error:** Only errors will be logged.
  - **Info:** Errors, warnings, and other informational messages will be logged.
  - **Debug:** A high level of detail will be used in the messages, primarily for debugging purposes.
  - **Default:** Info
- **Use for Reverse Replication:** Indicates whether this agent will be used for Reverse Replication; returns user input from the Publish instance to the Author instance

WARNING This account must have read access to all paths that will be replicated. In addition, you will want to create a new user for replication purposes and not dispose the 'admin' user for replication.

Transport Tab Configuration Parameters

- **URI:** This specifies the receiving servlet at the target location in particular. You can specify the host name (or alias) and context path to the target instance here. For example:
  - A default agent may replicate to http://localhost:4503/bin/receive?sling:authRequestLogin=1
  - A Dispatcher Flush Agent may replicate to http://localhost:80/dispatcher/invalidate.cache

The protocol specified here (HTTP or HTTPS) would determine the transport method.

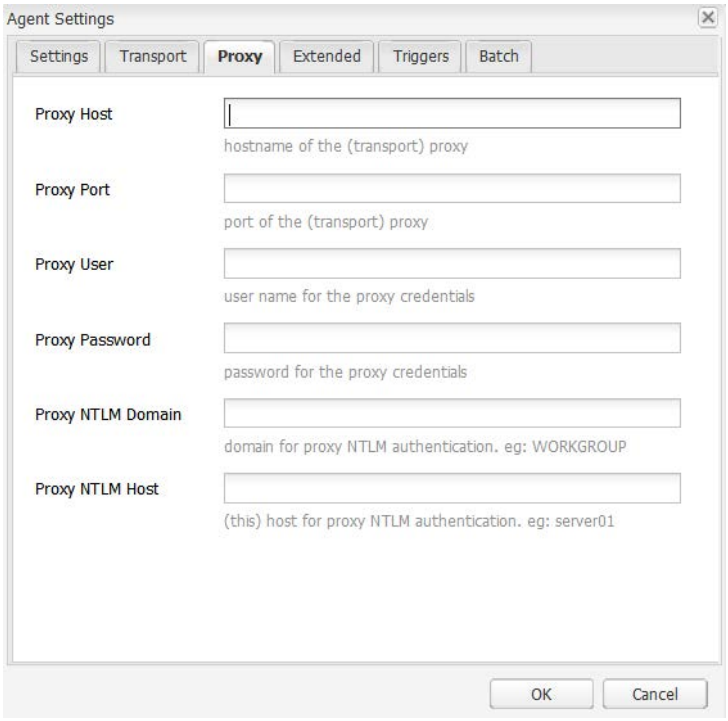


The Replication Agent Configuration Dialog Box: Transport Settings Tab

- **User:** User name of the account to be used for accessing the target
- **Password:** Password for the account to be used for accessing the target
- **NTLM Domain:** Domain for NTLM authentication
- **NTLM Host:** Host for NTLM authentication
- **SSL:** Enabled if you want self-certified SSL certificates to be accepted
- **Allow expired certs:** Enabled if you want expired SSL certificates to be accepted

Proxy Tab Configuration Parameters

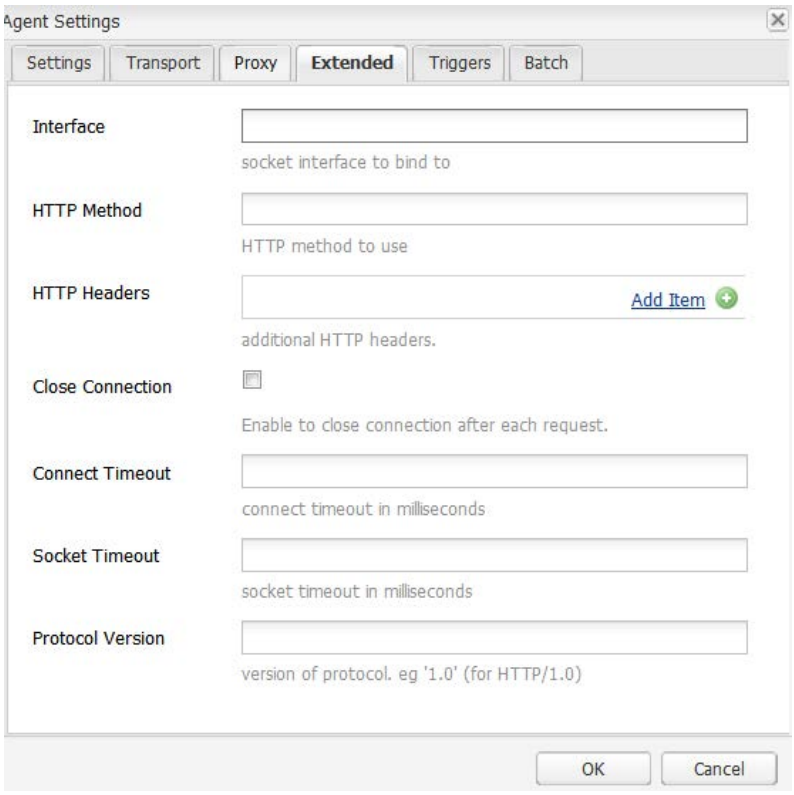
The following settings are only needed if a proxy is configured in the network.



The Replication Agent Configuration Dialog Box: Proxy Settings Tab

- **Proxy Host:** Host name of the proxy used for transport
- **Proxy Port:** Port of the proxy
- **Proxy User:** User name of the account to be used
- P**roxy Password:** Password of the account to be used
- **Proxy NTLM Domain:** The proxy NTLM domain
- **Proxy NTLM Host:** The proxy NTLM host

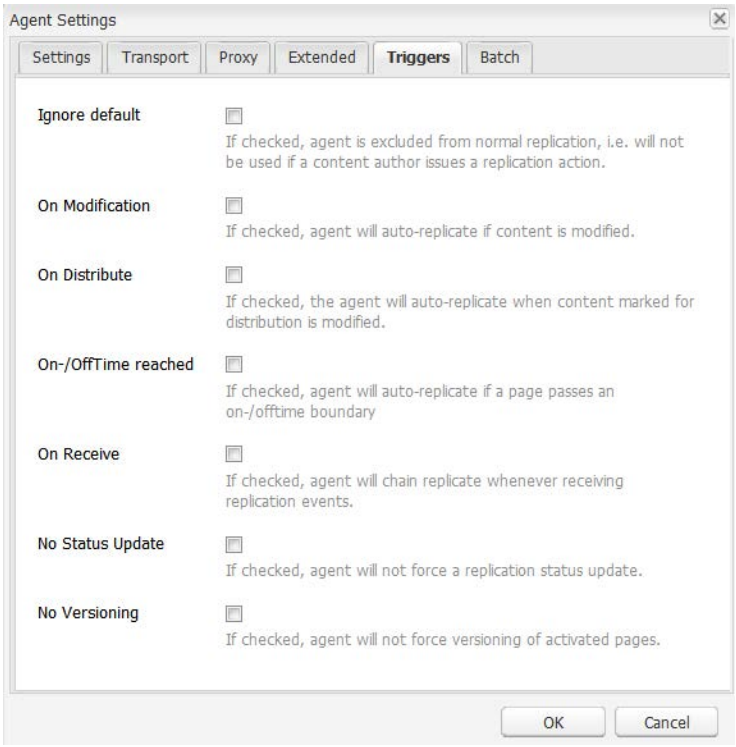
Extended Tab Configuration Parameters:



The Replication Agent Configuration Dialog Box: Extended Settings Tab

- **Interface:** Socket interface to bind to
- **HTTP Method:** HTTP method to use
- **HTTP Headers:** Used for Dispatcher Flush Agents and specify elements that must be flushed
  {action} indicates a replication action; {path} indicates a path.
- **Close Connection:** Close the connection after each request
- **Connect Timeout:** Timeout (in milliseconds) to be applied when trying to establish a connection
- **Socket Timeout:** Timeout (in milliseconds) to be applied when waiting for traffic after a connection has been established
- **Protocol Version:** Version of the protocol; for example, 1.0 for HTTP/1.0

Triggers Tab Configuration Parameters



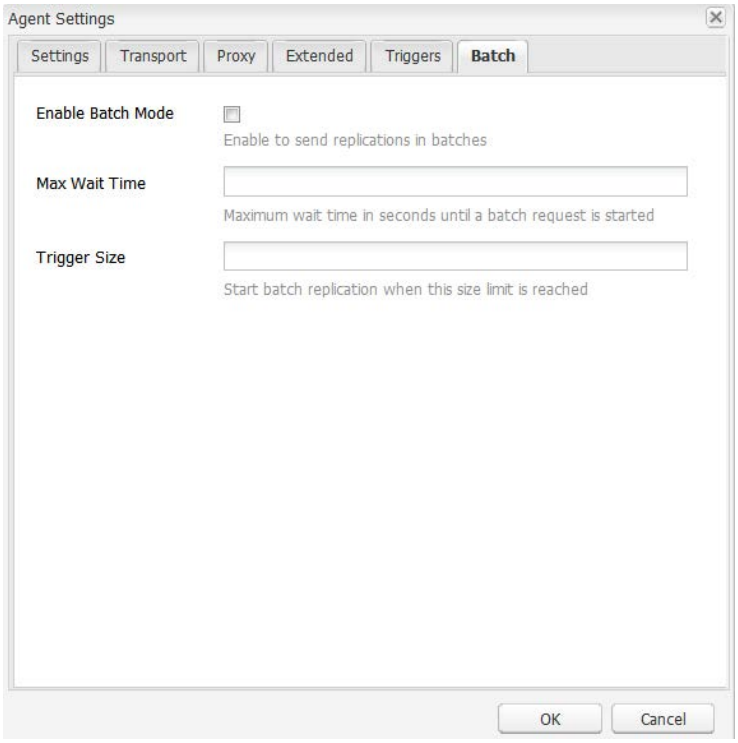
The Replication Agent Configuration Dialog Box: Triggers Settings Tab

These settings are used to define triggers for automated replication:

- **Ignore default:** If checked, the agent is excluded from default replication; this means it will not be used if a content author issues a replication action.
- **On Modification:** Here, a replication by this agent will be automatically triggered when a page is modified. This is mainly used for Dispatcher Flush Agents, but also for Reverse Replication.
- **On Distribute:** If checked, the agent will auto-replicate when content marked for distribution is modified.
- **On-/OffTime reached:** This will trigger automatic replication (to activate or deactivate a page as appropriate) when the on-times or off-times defined for a page occur. This is primarily used for Dispatcher Flush Agents.
- **On Receive**: If checked, the agent will chain replicate whenever receiving replication events.
- **No Status Update:** If checked, agent will not force a replication status update.
- **No Versioning:** If checked, agent will not force versioning of activated pages.

The purpose of the **Triggers** tab is to configure the agents to start replication automatically as soon as certain conditions occur.

Batch Tab Configuration Parameters



The Replication Agent Configuration Dialog Box - Batch Settings Tab

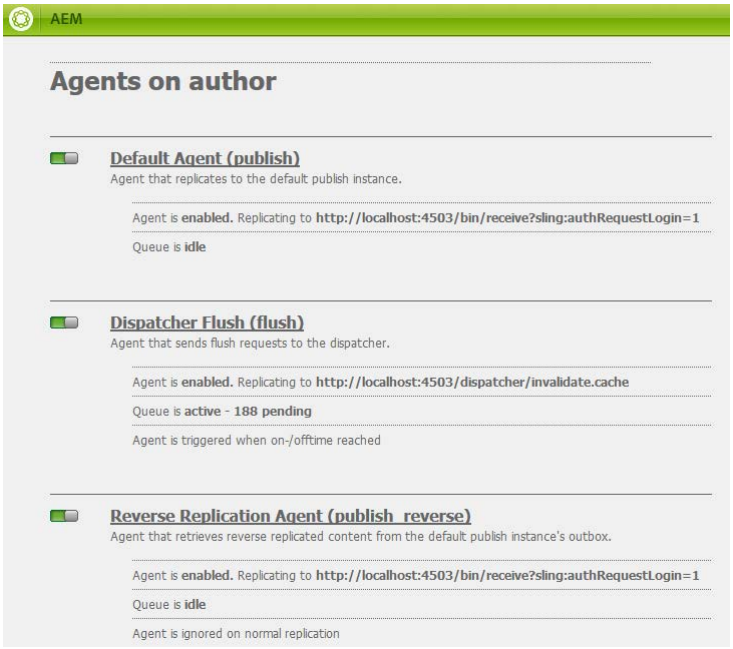
These settings are used to define batch mode for replication:

- **Enable Batch Mode:** Check this box to send replication in batches
- **Max Wait Time:** Maximum wait time in seconds until a batch request is started
- **Trigger Size:** Mention the size to start batch replication when this size limit is reached

How to Monitor Replication Agents

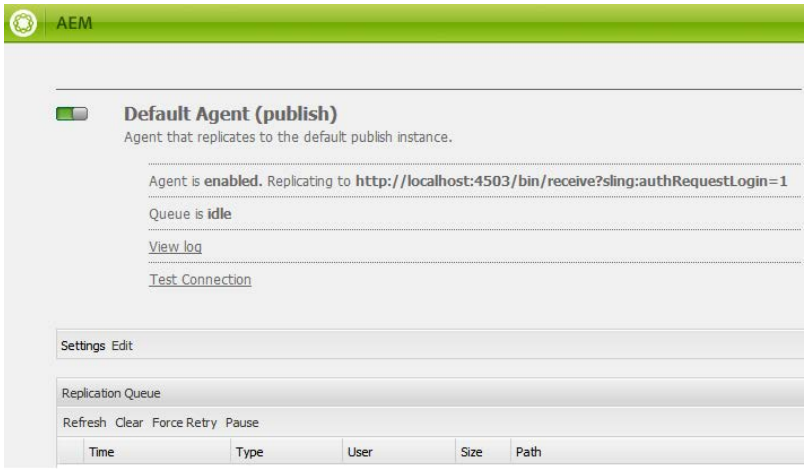
To monitor a Replication Agent:

1. Access **Tools** > **Operations** > **Replication** in the Rail or open http://localhost:4502/etc/replication.html:
2. Double-click the link to agents for the appropriate environment, for example, **Agents on author.** The resulting window shows an overview of all your Replication Agents for the Author instance, including their target and status:



Dialog Box of the Replication Agent

3. Click the appropriate agent name (which is a link) to show detailed information about that agent.



Dialog Box to Edit an Individual Replication Agent

Here, you can:

- See whether the agent is enabled
- See the target of any replications
- See whether the replication queue is currently active, and if so, any items in the queue
- Access the log of any actions by the Replication Agent
- Test the connection to the target instance

- Refresh, clear (selectively or entirely), or pause the display of queue entries
- Force a retry on any queue items if required

Exercise 7.1
Install Two Publish Instances to Configure Replication

To set up a Publish Instance on port 4503 of the desired host, you perform the same steps as in installing an author instance except that you create a directory named publish (instead of author) and you rename the cq-quickstart-6.0.jar file as cq-publish-4503.jar. You can select any port number.

For your test environment, you will install an additional Publish Instance. It will run on port 4505. Follow the same steps as described above or in Exercise 1 in chapter 1 (Installation) to set up a second Publish Instance. It is a common convention to use even port numbers for author instances and odd port numbers for Publish Instances.

Publish Instance Folder Structure

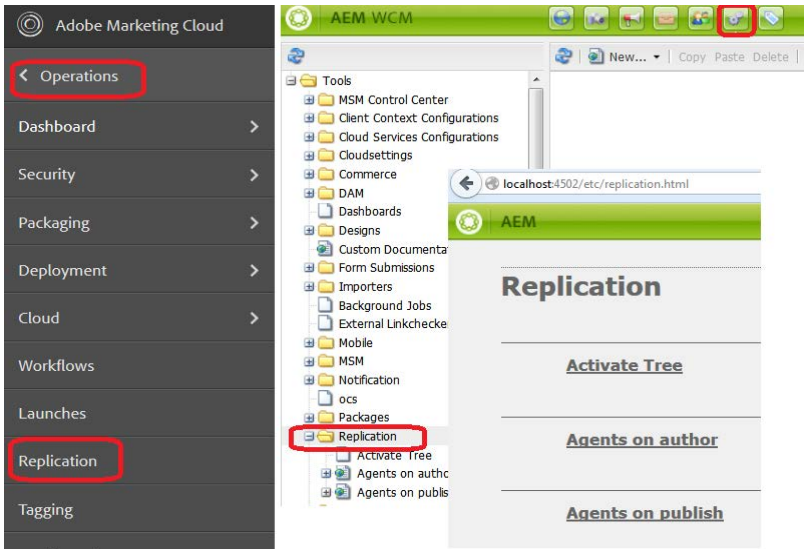
1. Windows: Create a folder structure on your file system where you will store, install, and start AEM:
   a. C:/adobe/AEM/publish1/cq-publish-4503.jar
   b. C:/adobe/AEM/publish2/cq-publish-4505.ja**r**
2. Mac OS or *x: Create the following structure commonly used for application installations:
   a. /opt/adobe/AEM/publish1/cq-publish-4503
   b. /opt/adobe/AEM/publish2/cq-publish-4505

   OR
   c. /Applications/AEM/ publish1/cq-publish-4503
   d. /Applications/AEM/ publish2/cq-publish-4505

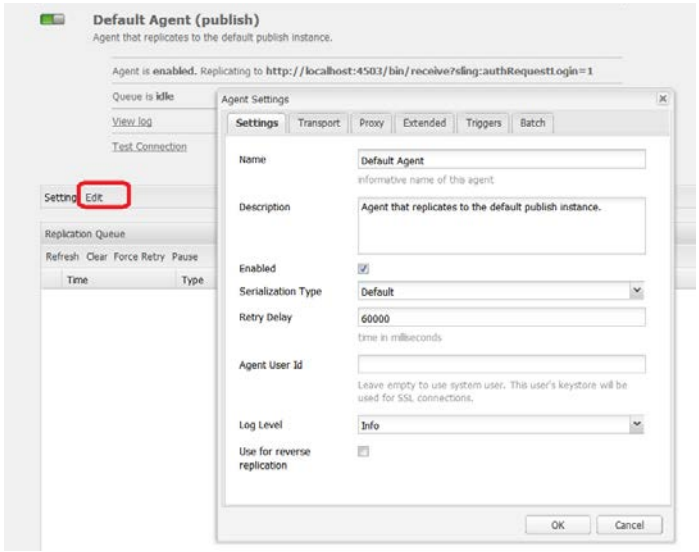
NOTE: The port number for publish1 is the default one which is already set up with the first Replication Agent, port number 4503. The second Publish Instance (publish2) should run on port 4505. Thus, jar files should be named appropriately. If using the startup scripts under <installation>/crx-quickstart/bin, update the CQ_PORT and CQ_RUNMODE variables accordingly.

Exercise 7.2
Configure and Access Replication Agents

1. Access the classic Tools console, click **Tools**, and open the replication page; alternatively, in the Touch UI by clicking the **Tools** link in the Rail, and by clicking **Operations** and **Replication** in the Rail. You can also directly open http://localhost:4502/etc/replication.html.



2. Click **Replication** (left pane) to open the folder.
3. Double-click **Agents on author** (either the left or from the right side link).
4. Click the default agent (which is a link) to show detailed information about that agent.
5. Click **Edit** to open the configuration dialog box:



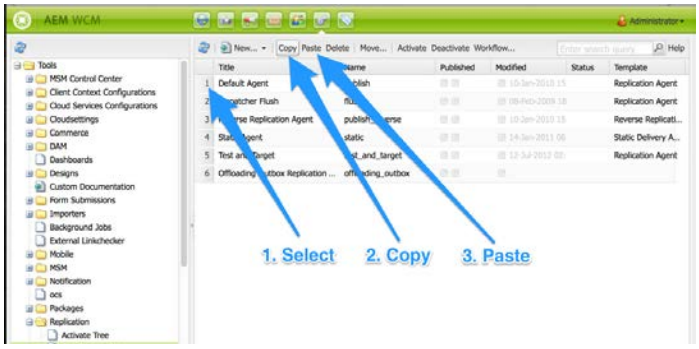
The Replication Agent Configuration Dialog Box

6. Click the **Transport** tab.
7. Make sure that the server and port specified in the URI are correct for the first Publish Instance.
8. Verify that the specified user and password are correct to access the first Publish Instance. You need to specify the credentials of the user waiting for data at the destination. In your case, these are the (default) credentials of the admin user on the Publish server.
9. Click **OK** to save the settings. Keep the default values in this out-of-the-box implementation on port 4503. If you are doing it for your own server, then replace the URI/user, password, and other values based on your environment.
10. Click **OK** to save the changes you have made on the agent **Settings** tab.

Now, you want to set up a second Delivery Agent, pointing to your second Publish Instance. You can create a new Delivery Agent from scratch and modify the settings accordingly. Similarly, you can create N number of agents as required.

Because your second Delivery Agent differs from the default one only by its destination instance, you can simply copy the Default Agent to reuse all other settings. To do this:

11. Return to the **Tools** pane at http://localhost:4502/miscadmin#/etc/replication/agents.author
12. Select the default agent.
13. Click **Copy** to copy the Default Agent and paste the new agent.



14. Open the new agent and click **Settings** to show the details for that agent.

NOTE: The agent names used here are for easy understanding. You should use names like 'reverse-localhost-4503' or 'replication-localhost-4502'. This will be easy to identify in a real implementation.

15. Give the new agent an appropriate name, for example, Publish2.

16. Click the **Transport** tab and set the URL to the correct values for the second Publish Instance; in your case, http://localhost:4505/bin/receive?sling:authRequestLogin=1. Also, make sure that the user and password are correct for the second Publish Instance.
17. Click **OK** to save the settings.

For a detailed description of Replication Agents, refer to the online documentation URL
http://dev.day.com/docs/en/cq/current/deploying/configuring_cq.html#Replication Agents.
Now that the second Replication Agent is set up, you can test page activation:

18. Go back to the AEM administrative console and open the **Websites** panel.
19. Browse to the **Geometrixx Demo Site** > **English** > **Products** page in the navigation panel.
20. Create a new page and open it.
21. Add some text, image, or any component you like on your page.
22. In the floating Sidekick panel, click the **Page** tab (the second tab).
23. Click **Activate Page** and confirm the activation.

Your page should now be accessible on both Publish Instances. Check it with the browser. Open the page on the Publish Instances, for example:
http://localhost:4503/content/geometrixx/en/products/<your page>.html
And
http://localhost:4505/content/geometrixx/en/products/<your page>.html

NOTE: If anything went wrong during the publishing process, check on the Author instance for the related Replication Agent log files using the configuration panel you used as you edited the settings.

Congratulations! You successfully created and configured two Replication Agents that will each replicate/activate to a Publish Instance. In addition, you successfully replicated to both Publish Instances.

Exercise 7.3
Activate a Tree

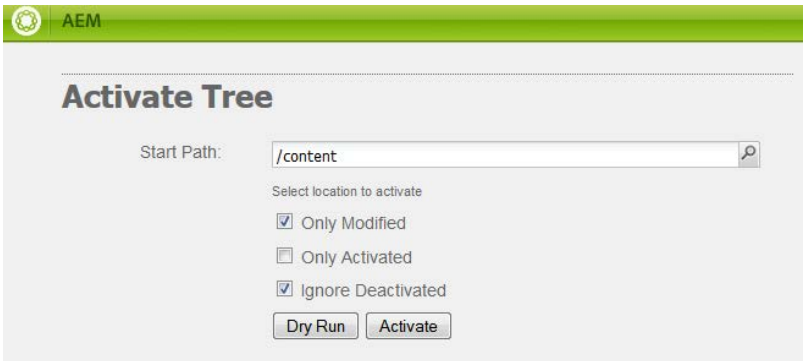
From the Websites tab, you can activate individual pages. When you have entered or updated a considerable number of content pages—all of which are residing under the same root page—it is easier to activate the entire tree in one action. You can also perform a dry run to emulate an activation and highlight which pages would be activated.

The following instructions explain how to browse the application/server interfaces associated with an AEM installation. This will enable you to use their administrative/configuration capabilities. To successfully complete and understand these instructions, you will need:

- A running AEM Author instance
- A running AEM Publish Instance (if not, available activation gets queued)

To activate a complete tree of your website:

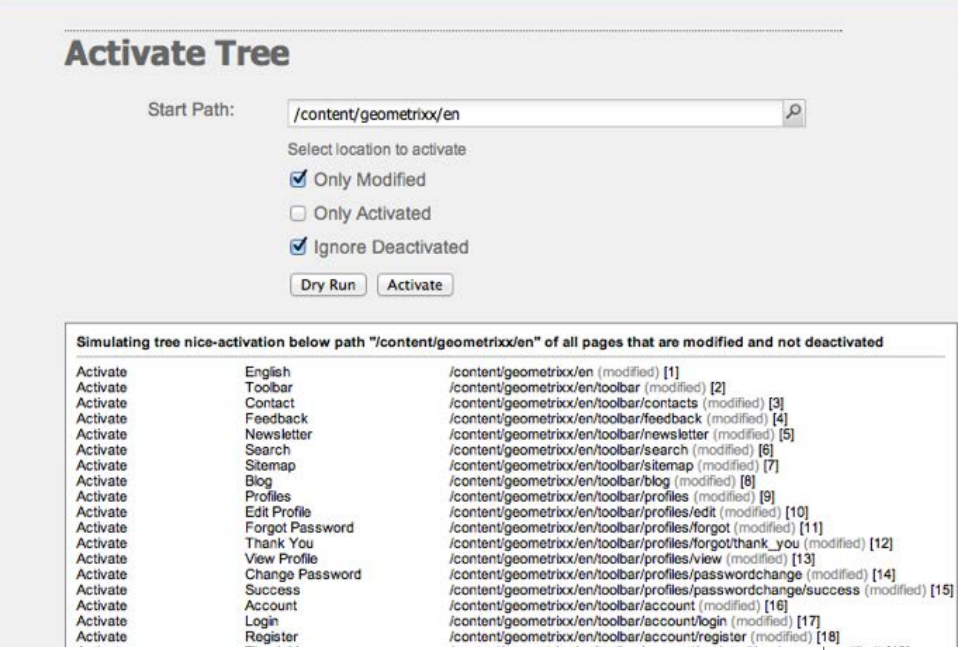
1. Access the **Tools** tab in the AEM admin console.
2. Click **Replication**. The folder expands.
3. Then, double-click **Activate Tree**.
   The following page opens.



Dialog Box to Define the Content Tree

4. Enter **/content/geometrixx/en** into the **Start Path** field or select the link from the drop-down selection. The Start Path field specifies the path to the root of the section you want to activate (publish). This page, and all pages underneath, will be considered for activation (or used in the emulation if a dry run is selected).
5. Choose the selection criteria as required:
   a. **Only Modified:** Only activate pages that have been modified.
   b. **Only Activated:** Only activate pages that have (already) been activated. This acts as a form of reactivation.
   c. **Ignore Deactivated:** Ignore any pages that have been deactivated.

6. Click **Dry Run** to check which pages would be activated. This is only an emulation; no pages will be activated.



A Dry Run Tree Activation

7. Click **Activate** to activate the tree.

Congratulations! You successfully activated a set (tree) of content pages.

Reverse Replication Agent

Replication Agents are used to propagate content from an Author instance to a Publish Instance. This kind of content replication occurs in one-way direction only. In a default environment, the Publish Instances are located behind a firewall that prevents Publish Instances to send the user-generated content on their instances directly to an Author instance. In order to propagate such a content to all Publish Instances, you need to pull that content from the Publish server and spread it through the regular mechanisms (default Replication Agents).

In this section, you set up two Reverse Replication Agents, one for each installed Publish Instance. The settings of a Reverse Replication Agent are similar to the ones of a Delivery Agent, which eases your setup work. In an out-of-the-box installation, there is already such an agent configured, pointing to a local Publish Instance's repository.

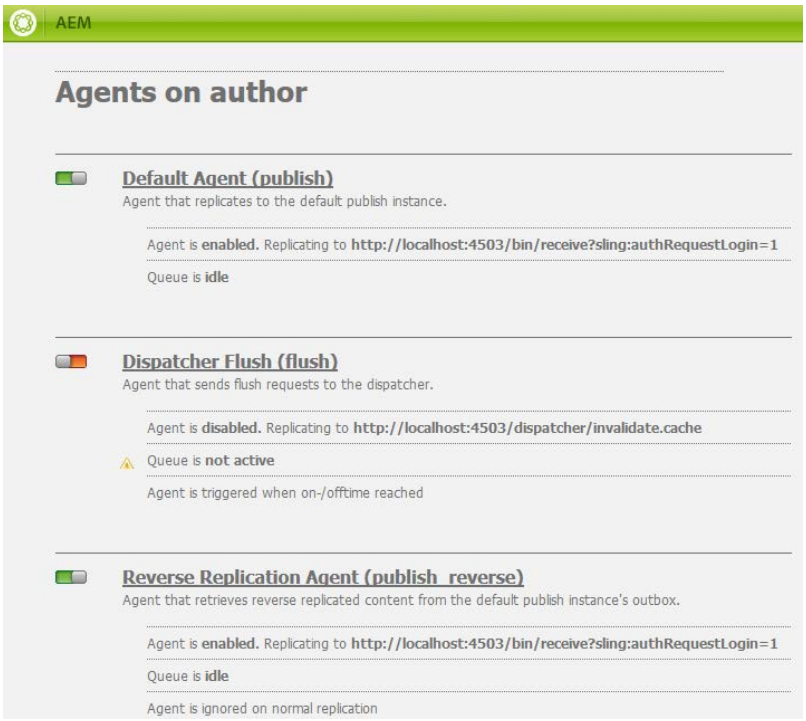
The following instructions explain how to create and configure Replication Agents. To successfully complete and understand these instructions, you will need:

- A running AEM Author instance
- Two running AEM Publish Instances

Exercise 7.4
Add a Reverse Replication Agent

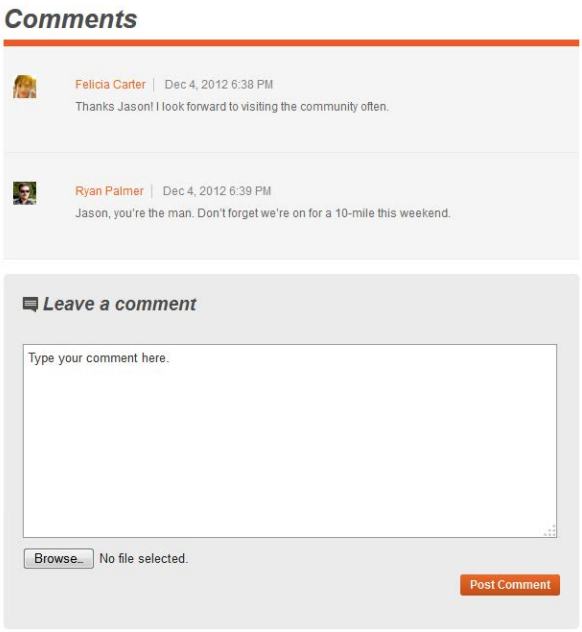
Access the Author agent's settings:

1. Access the Tools console of your Author instance.
2. Select the **Replication** folder in the left pane to expand.
3. Double-click the link to agents for the appropriate environment (from either the left or the right pane); for example, **Agents on author.** The resulting window shows an overview of all your Replication Agents for the Author instance, including their target and status:



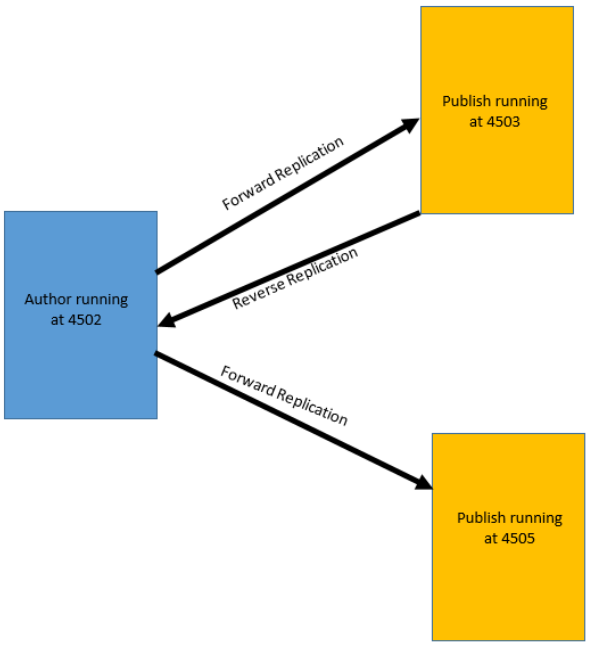
Available Replication Agents

4. Click the **Reverse Replication Agent (publish reverse)** link. The screen shows the status of the Reverse Replication Agent pointing to a default Publish instance running on port 4503. More setting details can be viewed or modified by clicking **Edit**. It is a good practice to test the connection to the destination point by following the link labeled **Test Connection**.
5. Now, open http://localhost:4503/content/geometrixx-outdoors/en/community/hiking.html to comment on the Publish instance is running on port 4503. Login as a user/admin to post a comment. After logging in, add a new comment on the page.



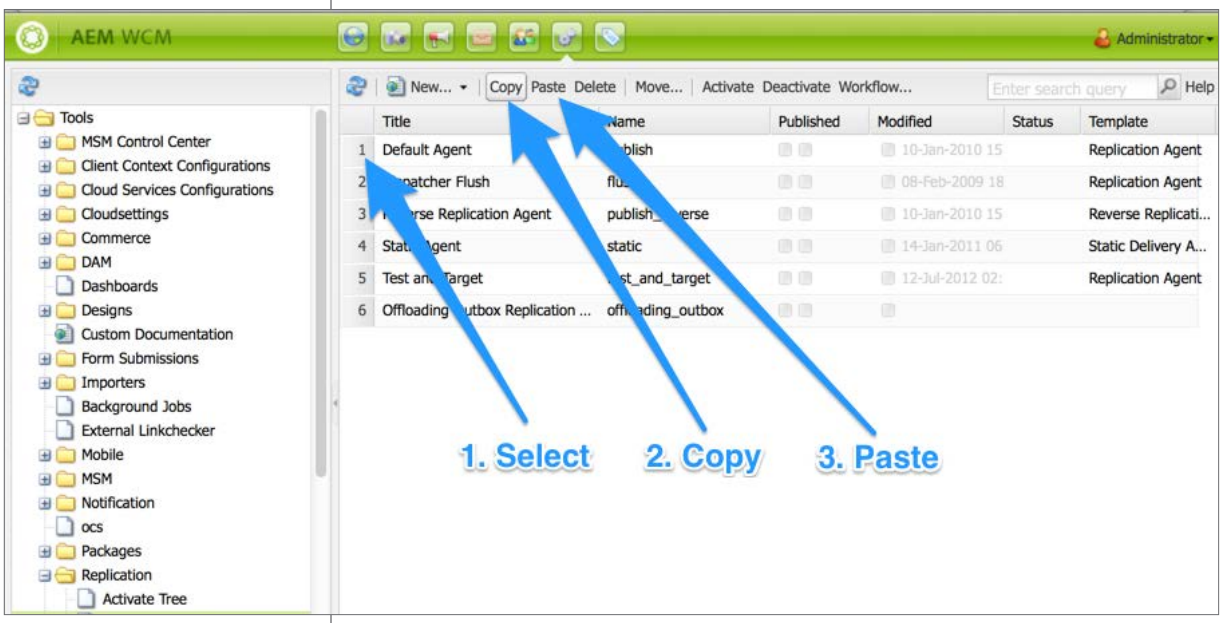
After few seconds (default: 5), notice that the comment appears on the author instance, hosted on port 4502, as default Reverse Replication Agent uses port 4503.

6. Now, comment on the page from the Publish instance running on port 4505 and observe that the comment is not replicated to the Author instance running on port 4502, as there is no Reverse Replication Agent configured as of now.



Now, create a new Agent for your second Publish instance running on port 4505.

1. Return to the **Tools** pane at http://localhost:4502/miscadmin#/etc/replication/agents.author.
2. In the Navigation pane (left tree), navigate to the node **Replication** > **Agents on author**.
3. Click **Copy** to copy the Default Reverse Replicator Agent (configured for port 4503 by default), paste the Reverse Replication Agent, and adopt its settings so that the agent works for Publish instance running on port 4505.



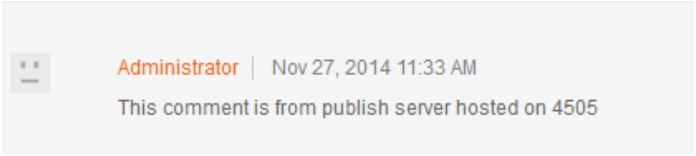
4. Rename the Reverse Replication Agent running on port 4503 as **Agent 1** and Reverse Replication Agent running on port 4505 as **Agent 2**.
   Ensure each agent is pointing to the desired Publish instance. Update the port numbers of the agents accordingly.

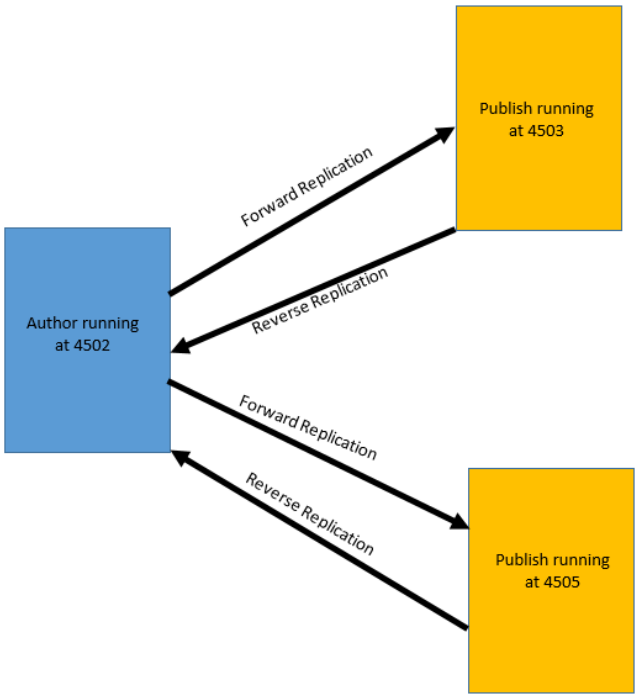
Test Reverse Replication Configured for Agent 2 Running on Port 4505

During a previous exercise, you replicated the entire Geometrixx > English website. The page, GeoBlog, is perfectly suitable for this test. It consists of a blog where every visitor can leave a comment. As a visitor is connected to one Publish instance at a time, the Reverse Replication Agent is responsible for retrieving the content (comment, reply), to be redistributed again to all the enclosed recipient publishers.
. Also test other forms, if desired.

1. On Publish instance running on port 4505, navigate to page: http://localhost:4505/content/geometrixx-outdoors/en/community/hiking.html. The Geometrixx Outdoors Hiking Community page appears.
2. Enter a new comment or reply to an existing one of your choice.

3. After few seconds (default: 5), your comment should be visible on the Author instance (4502) and on the other Publish instance (4503).





Congratulations! You successfully activated two Reverse Replication Agents and tested them.

Dispatcher

Goal

The Dispatcher is Adobe's caching and/or load balancing tool. Using Dispatcher also helps protect your application server from attack. Therefore, you can increase protection of your AEM instance by using Dispatcher with an industry-strength web server.

The AEM Dispatcher is a web server module. A version is available for Apache Web Server, Microsoft Internet Information Server (IIS), and iPlanet. While the installation of the AEM Dispatcher depends on the web server, the configuration is the same on all servers:

- Install the supported web server of your choice according to its own documentation.
- Install the AEM Dispatcher module appropriate to the chosen web server and configure the web server accordingly.
- Configure Dispatcher.
- Integrate with AEM to update the cache when the content is changed.

AEM Dispatcher Module Overview

- Static web servers are very simple, but fast.
- Dynamic websites can be slow.
- AEM Dispatcher converts dynamically published content (excluding personalized parts) to static HTML, serviced by a static web server.
- AEM Dispatcher is used with the web server.
- The environment is both fast and dynamic.
- Dispatcher is a web server plug-in available for Apache and IIS and is used to cache pages and static assets.
- It can be flushed by Replication Agents.
- Setting up a dispatcher will ensure that 90% content is delivered by Web server and only 10% by AEM.

Caching Methods

Content Updates

This method remove the pages that have changed in the repository. It will remove files directly associated with pages that have changed. This is implemented using a Dispatcher Flush Replication Agent on a Publish Instance.

Auto-Invalidation

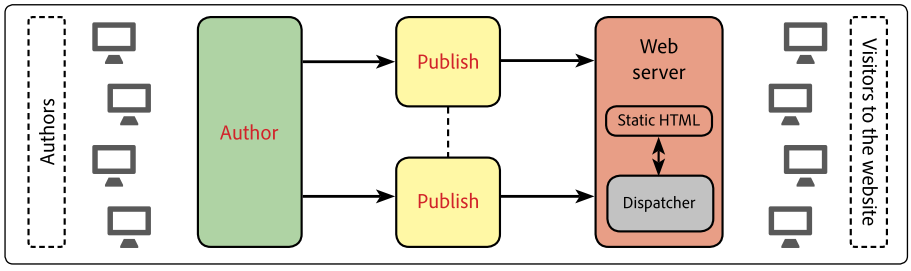
This method automatically invalidates parts of cache that may be out of date after an update in the repository. This will effectively flags relevant pages as being out of date; however, it does not delete any pages from the repository. This method is implemented using /statfiles setting in the dispatcher.any file.

Caching Choices

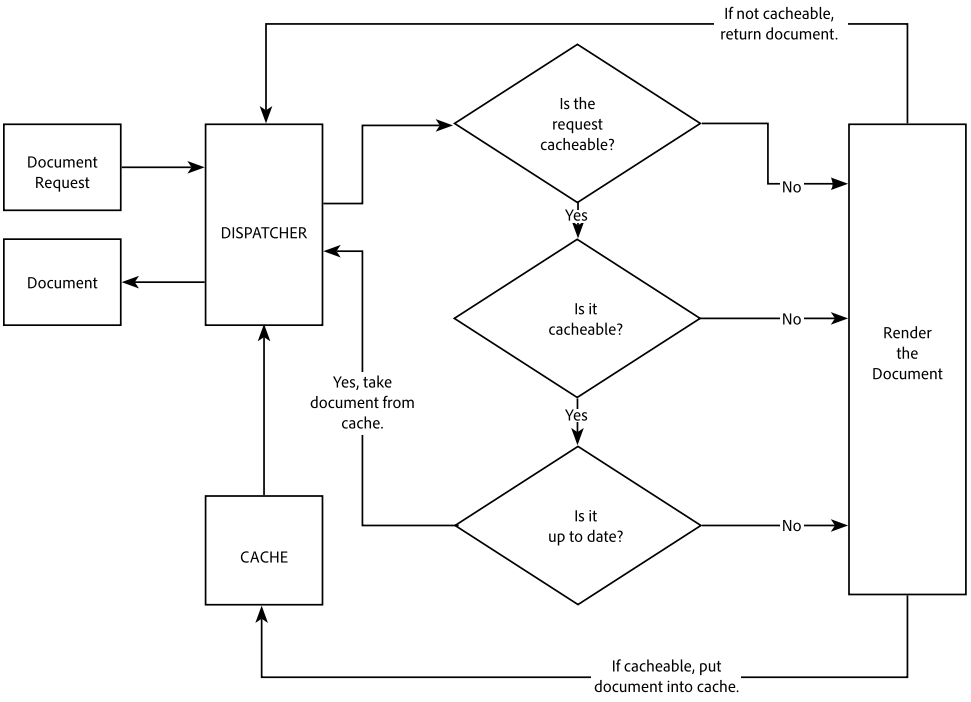
You can define which all documents should cached from AEM; if a document is not listed for caching, Dispatcher will request the same from AEM. Dispatcher will always request a document from AEM instance if:

- The HTTP method in the request is not 'GET'.
- The request URI contains a question mark (?). A question mark means that we have parameters along with the request that will change the response on the resulting document.
- The file extension is missing.
- The authentication header is set.

Dispatcher Module Architecture



Dispatcher Caching Algorithm



AEM Dispatcher and Load Balancing

Load balancing is the practice of distributing the computational load of the website across several instances of AEM.

Benefits of load balancing:

1. **Increased processing power**
   In practice, this means that Dispatcher shares document requests between several instances of AEM. Because each instance now has fewer documents to process, you have faster response times. The Dispatcher keeps internal statistics for each document category so it can estimate the load and distribute the queries efficiently.
2. **Increased fail-safe coverage**
   If Dispatcher does not receive responses from an instance, it will automatically relay requests to one of the other instance(s). Thus, if an instance becomes unavailable, the only effect is a slowdown of the site proportionate to the computational power lost. However, all services will continue. You can also manage different websites on the same static web server.

NOTE: While load balancing spreads the load efficiently, caching helps to reduce the load. Therefore, try to optimize caching and reduce the overall load before you set up load balancing. Good caching may increase the load balancer's performance or render load balancing unnecessary.

WARNING: While a single Dispatcher will usually be able to saturate the capacity of the available Publish instances, for some rare applications, it can make sense to additionally balance the load between two Dispatcher instances. Configurations with multiple Dispatchers need to be considered carefully since an additional Dispatcher will increase the load on the available Publish instances and can easily decrease performance in most applications.

Performance Statistics

The Dispatcher keeps internal statistics about how fast each instance of AEM processes a request. Based on this data, Dispatcher estimates which instance will provide the quickest response time when answering a request, and so it reserves the necessary computation time on that instance.

For most pages that use sticky connections, you have to switch off caching. Otherwise, the page looks the same to all users, regardless of the session content. For a few applications, it can be possible to use both sticky connections and caching, for example, if you display a form that writes data to the session.

Different types of requests may have differing average completion times, so Dispatcher allows you to specify document categories. These are then considered when computing the time estimates. For example, you can make a distinction between HTML pages and images, as the typical response times may well differ.

If you use an elaborate search function, you may create a new category for search queries. This helps Dispatcher send search queries to the instance that responds fastest. This prevents a slower instance from stalling when it receives several 'expensive' search queries, while the others receive the 'cheaper' requests.

Personalized Content (Sticky Connections)

Sticky connections ensure that documents for one user are all composed on the same instance of AEM. This is important if you use personalized pages and session data. The data is stored on the instance, so subsequent requests from the same user must return to that instance or the data is lost.

Because sticky connections restrict Dispatcher's ability to optimize the requests, you should use them only when needed. You can specify the folder that contains the 'sticky' documents, thus ensuring all documents in that folder are composed on the same instance for each user.

Add Dispatcher to the Apache Web Server

In this section, you will install Dispatcher into an Apache web server.

To successfully complete and understand these instructions, you will need:

- An Apache web server, up and running
- A running AEM Author instance
- A running AEM Publish Instance

The general outline for the Dispatcher setup is as follows:

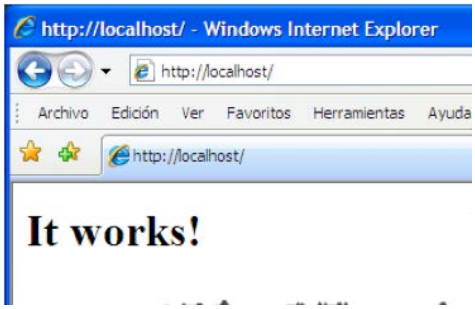
- Install Apache
- Install the Dispatcher Module
- Configure Apache (httpd.conf)
- Configure Dispatcher (dispatcher.any)
- Configure The Flush agent

Install Apache

For the Dispatcher setup; first, we would be installing the Apache web server. In the classroom we are installing it on our local machine for the understanding purpose. Your actual implementation may require you to install it on a server.

Exercise 9.1
Install the Apache Web Server

1. Make sure the Apache web server is up and running on your system.
2. To test whether Apache is already installed, access **http://localhost**. If the following screenshot appears, Apache is installed and running.



A new, working Apache installation will show this default page.

If not, follow these steps to install and start Apache:

3. Download the version that best suits you from http://httpd.apache.org/download.cgi, and follow the instructions to install and configure.
4. The binary download is sufficient for this exercise and it is provided in the USB drive content under the directory **<USB DRIVE>\distribution\apache**.

Install the Dispatcher Module

After installing Apache server, we need to install the Dispatcher module on the same system. The dispatcher module comes as binary form for every supported platform. This process will consist of following:

a. Downloading Dispatcher from adobeaemcloud.com or use a version from the USB contents.
b. Copy/paste dispatcher module (.dll or .so) files to the /modules folder in the Apache install directory.
c. Copy/paste dispatcher.any file to the /conf folder (same folder as the httpd.conf file)

Exercise 9.2
Install Dispatcher Into the Apache Web Server

1. Unzip the latest AEM Dispatcher build, appropriate for your operating system and web server, to a temporary directory. The Dispatcher files are located on the memory stick under **</USB>/distribution/dispatcher/Dispatcher Apache HTTP Server 2.2**.

2. Move the file, **disp_apache2.2.dll** (or the one corresponding to your Operating System version), into the directory **<apache_home>/modules**.
3. Move the Dispatcher configuration file **dispatcher.any** to the directory **<apache_home>/conf**.

Configure Apache

Configuring Apache web server for the Dispatcher includes loading appropriate Modules and other settings in **httpd.conf** file. To get the dispatcher module recognized by apache, you have to add it to the modules section of the httpd.conf of apache. These Modules and settings provide Dispatcher instructions so that it should behave in the desired manner.

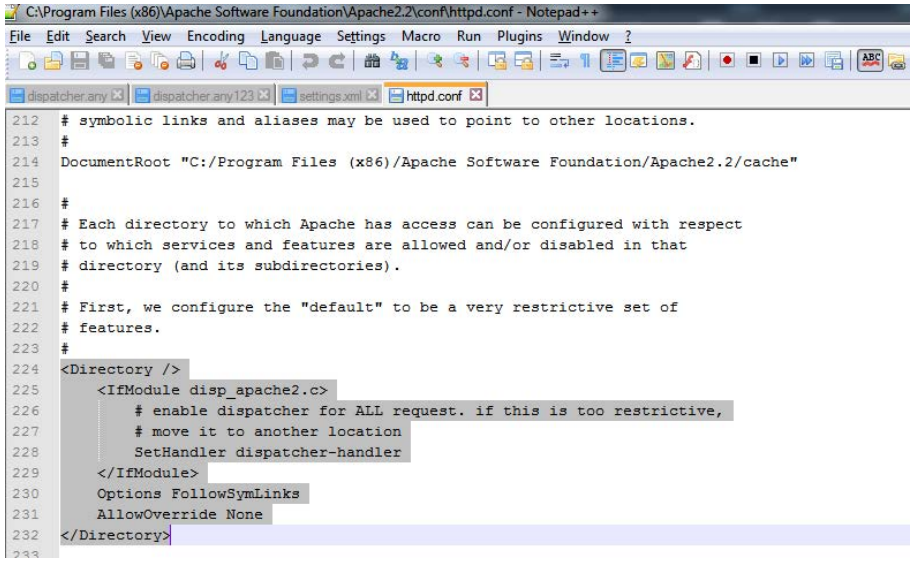
Exercise 9.3
Configure Apache Web Server

1. Navigate to **<APACHE_ROOT>/conf**, open the **httpd.conf** file, and add the required details as follows:
   a. Dispatcher-specific configuration entries, inside an ifModule, including:
      i. DispatcherConfig: Location and name of the configuration file, for example, **'conf/dispatcher.any'**.
      ii. DispatcherLog: Location and name of the log file, for example, **'logs/dispatcher.log'**.
      iii. DispatcherLogLevel: Log level of the log file, for example, **'DispatcherLogLevel 3'**.
      iv. DispatcherNoServerHeader: Indicates that the Apache server header should be used.
      v. DispatcherDeclineRoot: Defines whether to decline requests to root.
      vi. DispatcherUseProcesseduRL: Defines whether to use the pre-processed (by Apache web server) URL or the original URL.
      vii. DispatcherPassError: Defines how to support 40x error codes for Error Document Handling.
      viii. Load Module statement to load the Dispatcher module on startup. To load the module, open **conf/httpd.conf** file and load the module statement by adding the following:
      ```
      LoadModule dispatcher_module modules/disp_apache2.2.dll
      ```
      ix. SetHandler to activate Dispatcher.LoadModule. This is placed in the first <Directory> section after the LoadModule section. The SetHandler statement that you will be using configures Dispatcher to handle requests for the complete website.
      You must add a **SetHandler** statement to the context of your configuration (<Directory>, <Location>) for Dispatcher to handle incoming requests. The mode_mime module is used to assign content metadata to the content selected for an HTTP response. When 'on', the ModMimeUsePathInfo parameter specifies that mode_mime is to determine the content type based on the complete URL; this means that virtual resources will have meta information

applied based on their extension.

Make the first <Directory> element after the LoadModule section look like the following, to configure Dispatcher to handle requests for the complete website:

```
<Directory />
  <IfModule disp_apache2.c>
           SetHandler dispatcher-handler
       ModMimeUsePathInfo On
  </IfModule>
  Options FollowSymLinks
  AllowOverride None
</Directory>
```

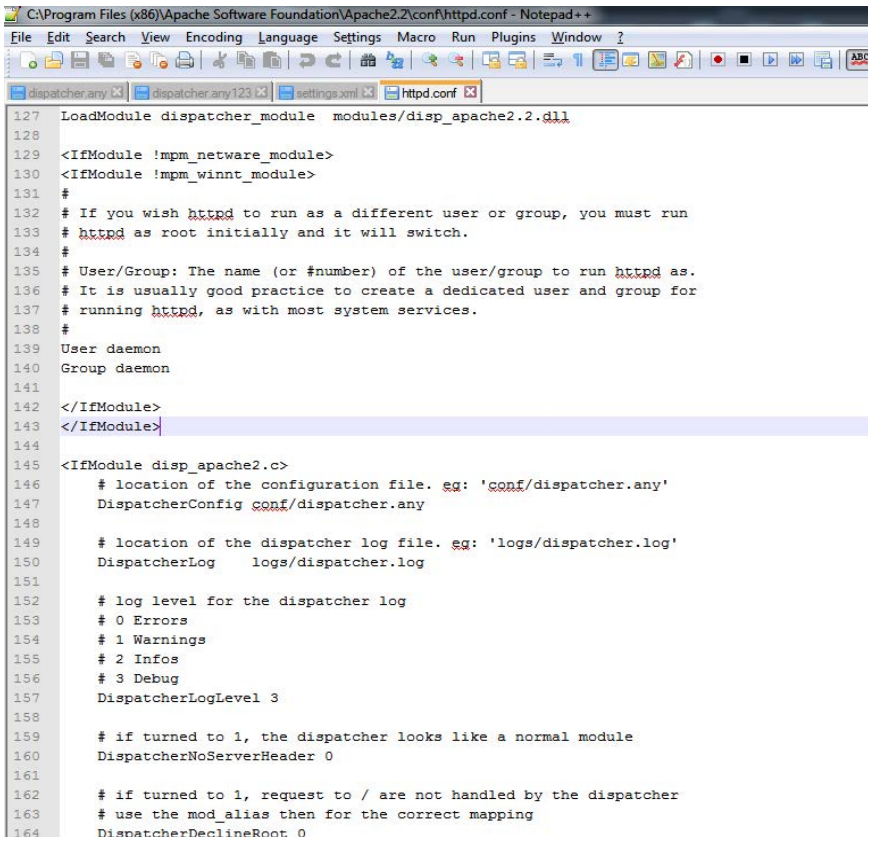


x. We have to check If 'MultiViews' is used in any of the <Directory> sections then we have to comment it out.

2. Create a folder named **cache** inside **C:\Program Files (x86)\Apache Software Foundation\Apache<version number>**.

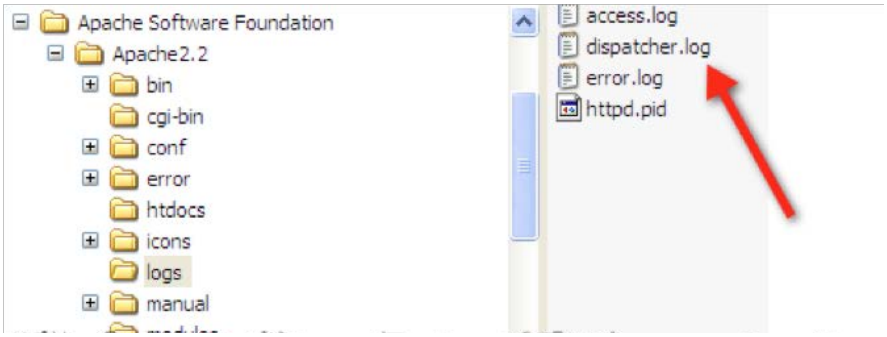
3. To complete step 1, add the following text to the **httpd.conf** file at the end of the LoadModule section, or just use the sample file containing all the configuration for Apache 2.2, which is available in USB content at: **<USB>\distribution\dispatcher\httpd.conf**

```
LoadModule dispatcher_module modules/disp_apache2.2.dll
<IfModule disp_apache2.c>
       #location of the configuration file.  e.g., “conf/dispatcher.any”
          DispatcherConfig conf/dispatcher.any
       #location of the dispatcher log file
          DispatcherLog logs/dispatcher.log
       #log level for the dispatcher log file, e.g., “logs/dispatcher.log”
       #0 Errors
       #1 Warnings
       #2 Infos
       #3 Debug
         DispatcherLogLevel 3
       #must be 1, so that the dispatcher looks like a regular module
       DispatcherNoServerHeader 1
   #if turned to 1, requests to “/” are not handled by the dispatcher
       #if turned to 1, use mod_alias for the correct mapping
         DispatcherDeclineRoot 0
       # if turned to 1, use the URL already processed
       #by the handlers that precede the dispatcher
         DispatcherUseProcessedURL 1
       #if 0, dispatcher spools all error responses to the client
         DispatcherPassError 0
</IfModule>
```



4. Restart the Apache web server.

5. You can see that Dispatcher has been installed successfully. Observe the log file **dispatcher.log** created as per our configuration.



Congratulations! You successfully integrated Dispatcher with the Apache web server.

Configure Dispatcher

Goal

Now that you have integrated the AEM Dispatcher with the web server, you must configure Dispatcher so that it can find its associated Publish instances, knows which pages to cache, and where to cache them.

In this chapter, you will configure the AEM Dispatcher with appropriate settings to cache pages as desired, and you will define a Dispatcher Flush Agent to 'invalidate' the cache in response to content update.

By default, the AEM Dispatcher configuration is stored in a file named dispatcher.any. You can change the name and location of the configuration file. If you do so, make sure to configure the module with the new name and location. The dispatcher.any file is independent of the web server and operating system, so the following instructions are appropriate to both IIS and Apache. The only difference between the two configurations is the usage of the property /homepage, which is used only by IIS.

The configuration file contains a series of single-valued or multi-valued properties that control the behavior of Dispatcher:

- Property names are prefixed with a forward slash (/).
- Multi-valued properties enclose child items using braces ({ }).

The following is an example Dispatcher configuration:

```
# name of the dispatcher
/name "internet-server"
# each farm configures a set off (loadbalanced) renders
/farms
  {
  # first farm entry (label is not important, just for your
    convenience)
  /website
    {
    /clientheaders
      {
      # List of headers that are passed on
```

```
    }
  /virtualhosts
    {
    # List of URLs for this Web site
    }
  /sessionmanagement
    {
    # settings for user authentication
    }
  /renders
    {
    # List of AEM instances that render the documents
    }
  /filter
    {
    # List of filters
    }
  /cache
    {
    # Cache configuration
    /rules
      {
      # List of cachable documents
      }
    /invalidate
      {
      # List of auto-invalidated documents
      }
    }
  /statistics
    {
    /categories
      {
      # The document categories that are used for load balancing
estimates
      }
    }
  /stickyConnectionsFor “/myFolder”
  /health_check
    {
    # Page gets contacted when an instance returns a 500
    }
  /retryDelay “1”
  /numberOfRetries “5”
  /unavailablePenalty “1”
  }
}
```

You should check and verify the list of client headers in the clientheaders section. The following code already exists in the clientheaders section and you do not need to do anything in case it is not required/valid for your implementation.

```
...
# each farm configure a set off (LoadBalanced) renders
/forms
   {  |
    # first farm entry (label is not important, just for your convenience)
    /website
       {
       # client headers which should be passed through to the reader instances
       /clientheaders
                {
        “referer”
        “user-agent”
        “from”
        “content-type”
        “content-length”
        “accept-charset”
        “accept-encoding”
        “accept-language”
        “accept”
        “host”
        “if-match”
        “if-none-match”
        “if-range”
        “if-unmodified-since”
        “max-forwards”
        “proxy-authorization”
        “proxy-connection”
        “range”
        “cookie”
        “cq-action”
        “cq-handle”
        “handle”
        “action”
        “cqstats”
        }
. . . .
```

The **virtualhosts** section is a list of all hostname/URI combinations that Dispatcher accepts for this website. You can also use the asterisk (*) character as a wildcard. Again, this information may not be correct in your implementation. Therefore, you need to take care of the Virtualhost entries. Check the hostname/URI in your actual implementation while doing it for a client/customer.

Adapt the virtualhosts property.

```
...
/farms
   {
   # first farm entry (label is not important, just for your
    convenience)
                  /website
                 {
                       ...
                        /virtualhosts
              {
              # entries will be compared against the “host”
    request header
              # example: www.company.com
              # example: intranet.*
              #example: mysite.*/mysite/*
              ...
              }
                        ...
```

The **/cache** section specifies general aspects of how and where the AEM Dispatcher caches documents. Included in the /cache section are:
/docroot: This link points to the document root of the web server.

/statfile and /statfileslevel: This defines which parts of the website tree are invalidated when pages are activated.

/allowAuthorized: This specifies whether requests (pages) that carry an authentication header are cached.

/rules: This is a list of cacheable documents determines which documents are cached.

/invalidate: This is a list of cacheable documents; it determines which documents are cached.
/docroot: This is a link, which points to the document root of the web server. This is where the AEM Dispatcher stores the cached documents, and this is where the web server looks for them. If you use multiple render farms, you have to define a different document root on the web server for each farm and specify the corresponding link here.

Define the location of the web server cache to the AEM Dispatcher.

```
   {
      # the cacheroot must be equal to the document of web
   server
      # /docroot “C:/Inetpub/wwwroot”
        /docroot “<Apache_document_root>”
   ...
```

Understanding the dispatcher.any File contents

1. Make sure the **/farms** section matches your infrastructure. The /farms section defines a list of farms or websites. Each /farms section defines:
   a. a set of load-balanced renderers
   b. the IP addresses and ports of the publish instances to serve and cache content from
   c. further characteristics, including where to cache files and what to cache

For each farm, you can specify separate caching and rendering parameters, some of which have sub-parameters:

Property name	Description
/homepage	Default homepage (optional, IIS only)
/clientheaders	The headers from the client HTTP request to pass through
/virtualhosts	The virtual hosts for this farm
/sessionmanagement	Support for session management and authentication
/renders	The servers that provide rendered pages (typically AEM publish instances)
/filter	Defines the URLs to which Dispatcher enables access
/propagateSyndPost	Support for the forwarding of syndication requests
/cache	Configures caching behavior
/statistics	Defines statistical categories for load-balancing calculations
/stickyConnectionsFor	The folder that contains sticky documents
/health_check	The URL to use to determine server availability
/retryDelay	The delay before retrying a failed connection
/unavailablePenalty	Penalties that affect statistics for load-balancing calculations
/failover	Resend requests to different renders when the original request fails.

If required, you can include other files that contribute to the configuration:

- If your configuration file is large, you can split it into several smaller files (that are easier to manage), and then include those. You can include files that are automatically generated.

For example, to include the file **myFarm.any** in the /farms configuration, use the following code:

```
/farms
{
$include “myFarm.any”
}
```

NOTE: If you add your own header, you have to add all default headers to the configuration section.

NOTE: Adobe best practices suggest that you let the web server handle virtual hosting

The **clientheaders** section defines a list of all HTTP headers passed from the client to the AEM instance. By default, the AEM Dispatcher forwards the standard HTTP headers to the AEM instance. In some instances, you might want to:

- Add headers: for example, custom headers
- Remove headers: for example, authentication headers that are only relevant to the web server

2. The **renders** section defines where the AEM Dispatcher will allocate requests to render a document. The Dispatcher will load balance among all the defined renders. Adapt the **renders** property for the available publish instances. This will forward all requests to the specified Publish Instance(s) (called renders in Dispatcher configuration). You can define several renders within a farm for load balancing

3. When configuring Dispatcher, you should restrict external access as far as possible, for example, only the following are available to external visitors:
   a. /content
   b. Miscellaneous content such as designs and client libraries; for example:
      i. /etc/designs/default*
      ii. /etc/designs/mydesign*

The following is an example /filter section, from the dispatcher.any file. The recommended principle is to deny access to everything, and then allow access to specific (limited) elements:

```
# only handle the requests in the following acl. default is
    'none'
    # the glob pattern is matched against the first request
    line
    /filter
      {
      # deny everything and allow specific entries
      /0001 { /type "deny"  /glob "*" }

      # open consoles
#     /0012 { /type "allow" /glob "* /crx/*"    }
# allow content repository
#     /0013 { /type "allow" /glob "* /system/*" }
# allow OSGi console

      # allow non-public content directories
#     /0021 { /type "allow" /glob "* /apps/*"   }
# allow apps access
#     /0022 { /type "allow" /glob "* /bin/*"    }
      /0023 { /type "allow" /glob "* /content*" }
# disable this rule to allow mapped content only

#     /0024 { /type "allow" /glob "* /libs/*"   }
#     /0025 { /type "deny"  /glob "* /libs/shindig/proxy*" }
```

```
# if you enable /libs close access to proxy

#    /0026 { /type “allow” /glob “* /home/*”   }
#    /0027 { /type “allow” /glob “* /tmp/*”    }
#    /0028 { /type “allow” /glob “* /var/*”    }

# enable dynamic media
     /0030 { /type “allow” /glob “* /is/image*”   }
     /0031 { /type “allow” /glob “* /is/content*”   }

# enable specific mime types in non-public content directories
     /0041 { /type “allow” /glob “* *.css *”   }
# enable css
     /0042 { /type “allow” /glob “* *.gif *”   }
# enable gifs
     /0043 { /type “allow” /glob “* *.ico *”   }
# enable icos
     /0044 { /type “allow” /glob “* *.js *”    }
# enable javascript
     /0045 { /type “allow” /glob “* *.png *”   }
# enable png
     /0046 { /type “allow” /glob “* *.swf *”   }
# enable flash
     /0047 { /type “allow” /glob “* *.svg *”   }
# enable SVG
     /0048 { /type “allow” /glob “* *.woff *”  }
# enable woff
     /0049 { /type “allow” /glob “* *.ttf *”   }
# enable ttf
     /0050 { /type “allow” /glob “* *.eot *”   }
# enable eot
     /0051 { /type “allow” /glob “* *.jpg *”   }
# enable jpg

# enable features
     /0061 { /type “allow” /glob “POST /content/[.]*.form.
    html” }
# allow POSTs to form selectors under content
     /0062 { /type “allow” /glob “* /libs/cq/
    personalization/*” }
# enable personalization
     /0063 { /type “allow” /glob “POST /content/[.]*.commerce.
    cart.json” }
# allow POSTs to update the shopping cart
# deny content grabbing
     /0081 { /type “deny”  /glob “GET *.infinity.json*” }
     /0082 { /type “deny”  /glob “GET *.tidy.json*”     }
```

```
        /0083 { /type “deny”  /glob “GET *.sysview.xml*”   }
        /0084 { /type “deny”  /glob “GET *.docview.json*”  }
        /0085 { /type “deny”  /glob “GET *.docview.xml*”   }
        /0086 { /type “deny”  /glob “GET *.*[0-9].json*”   }
        /0087 { /type “deny”  /glob “GET *.feed.xml*”      }
#       /0088 { /type “allow” /glob “GET *.1.json*”        }
# allow one-level json requests

        # deny query
        # This is only required if dispatching for CQ 5.5 or
      older
        /0090 { /type “deny”  /glob “* *.query*” }
        }
```

NOTE: Adobe best practices suggest that you deny access to /libs, /etc, /crx, /admin, /var, /tmp, /home, /apps, and any other URIs that should not be accessible by an unauthorized user. See the Security Checklist for further considerations when restricting access using the AEM Dispatcher.

Exercise 10.1
Configure the dispatcher.any File

1. Open the **dispatcher.any** file with the text editor of your choice.
2. Update **/rend01** to your publish server address.
3. Add **/rend02** if you implement two publish instances.
4. Update **/docroot** to point to the same location as DocumentRoot in the **httpd.conf** file

By default, requests that carry an authentication header are not cached. This is because authentication is not checked when a document is returned from the cache, therefore the document would be displayed for a user who does not have the necessary rights. However, in some setups, it can be permissible to cache authenticated documents.
Configuration of the AEM Dispatcher is not yet complete, but at this point, you can test the configuration of Dispatcher with the web server.

5. Save dispatcher.any changes.
6. Restart the web server.
7. Access the Geometrixx website using the following URLs:

Author instance: http://localhost:4502/content/geometrixx/en/products/circle.html
Publish instance: http://localhost:4503/content/geometrixx/en/products/circle.html
Web server/Dispatcher: http://localhost/content/geometrixx/en/products/circle.html

Test Your Configuration

You have not yet completed the AEM Dispatcher configuration. For example, you still have not defined a way for the AEM Dispatcher to invalidate cached pages. However, the cached pages should be showing up in the web server's document root.

If you now navigate to the web server document root directory on the disk, you should see directories and files appear as the AEM Dispatcher and web server begin to cache pages.

If you continue to navigate through the Geometrixx website using the web server (default port 80), you should see the set of cached pages grow.

Exploring more of dispatcher.any File contents

Now that you know you have a good connection between the web server, AEM Dispatcher, and Publish instance, you can complete the configuration of the AEM Dispatcher.

8. The **/statfile** link points to the 'statfile,' which the AEM Dispatcher uses to keep track of the last content update. This can be any file on the web server. The file itself is empty; only the timestamp is used.

NOTE: The statfil is a simple hidden text file. When a page is replicated, the Dispatcher Flush Agent sends a notification to the Dispatcher. The Dispatcher responds to the notification by updating the timestamp of the statfile. Cached files are served only if they are newer than the timestamp of the nearest statfile.

The **/statfileslevel** link causes statfiles to be created in all folders down to the level you specify. Use this if you only want to invalidate sections of the cache, not the entire cache.

For a default structure with language folders (for example, the English site under /content/myWebsite/en), use /statfileslevel = "2" to invalidate per language. This means that when the AEM Dispatcher invalidates the English part of the website, other sections are not affected (for example, the German and French sections).

You may use /statfile or /statfileslevel, but not both.

Set the **/statfileslevel** property.

```
/cache
  {
  /docroot "C:/apache/htdocs"
  /statfileslevel "2"
  ...
```

9. The **/rules** property defines which documents are cached, though the Dispatcher never caches a document in the following circumstances:
   a. If the HTTP method is not GET. Other common methods are POST for form data and HEAD for the HTTP header.

b. If the request URI contains a question mark (?).
   This usually indicates a dynamic page, such as a search result that does not need to be cached. The file extension is missing.
c. The web server needs the extension to determine the document type (the MIME-type). The authentication header is set (this can be configured).

If you do not have dynamic pages (beyond those already excluded by the above rules), you can let the AEM Dispatcher cache everything.

10. Define the list of cacheable documents:

```
/cache
  {
  /docroot “C:/apache/htdocs”
  /statfileslevel  “2”
  /allowAuthorized “0”
  /rules
   {
     /0000
      {
      /glob “*”
      /type “allow”
      }
    /0001
      {
      /glob “/en/news/*”
      /type “deny”
      }
   /0002
    {
    /glob “*/private/*”
    /type “deny”
    }
   }
  ...
```

11. The **/invalidate** section defines a list of all documents that are automatically rendered invalid after a content update.

With **auto-invalidation**, the Dispatcher does not physically delete the pages after a content update, but checks them for validity when they are next requested. Documents in the cache that are not auto-invalidated will remain in the cache until they are deleted by a content update.

Auto-invalidate is typically used for HTML pages. As these often contain navigation elements and links to other pages, it is hard to determine whether a page is affected by a content update. To be safe, you usually auto-invalidate all HTML pages.

If you offer automatically generated PDF and zip files for download, you might also have to auto-invalidate these.

Set the file type list of documents that are automatically invalidated after any activation:

/cache

```
{...
   /invalidate
    {
     /0000
     {
       /glob “*”
       /type “deny”
     }
     /0001
     {
      /glob “*.html”
      /type “allow”
     }
     /0002
     {
     /glob “*.zip”
     /type “allow”
     }
     /0003
     {
     /glob “*.pdf”
     /type “allow”
     } }...
```

Exercise 10.2
Complete the Dispatcher Configuration

1. Open the **dispatcher.any** file with the text editor of your choice.
2. Uncomment **/statfilelevel** “2”
3. Uncomment **/allowAuthorized** “1”

```
/cache
  {
  /docroot “C:/apache/htdocs”
  /statfileslevel  “2”
  /allowAuthorized “1”
  ...
```

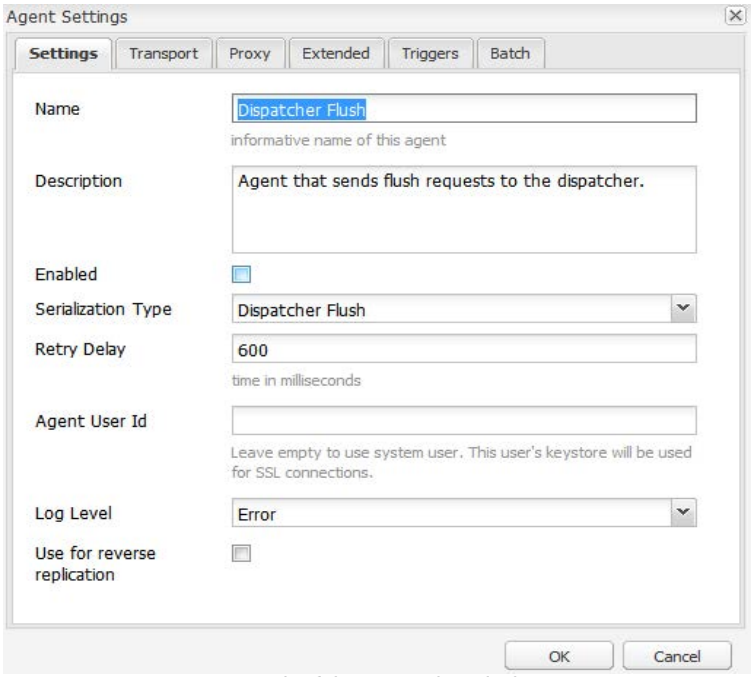
4. Save **dispatcher.any** changes.

Congratulations! Now you have a fully configured Dispatcher. The accessed web documents are being cached, but newly activated documents will not replace the cached documents until you have configured a Dispatcher Flush Agent that will invalidate the cache.

Exercise 10.3
Configure the Dispatcher Flush Agent

In cases where there are multiple Publish Instances, the flushing/invalidating of the cache by the AEM Dispatcher is controlled by a Replication Agent, running on the publish instance. However, the configuration is made on the authoring instance and then transferred to the Publish Instance(s) by activating the agent:

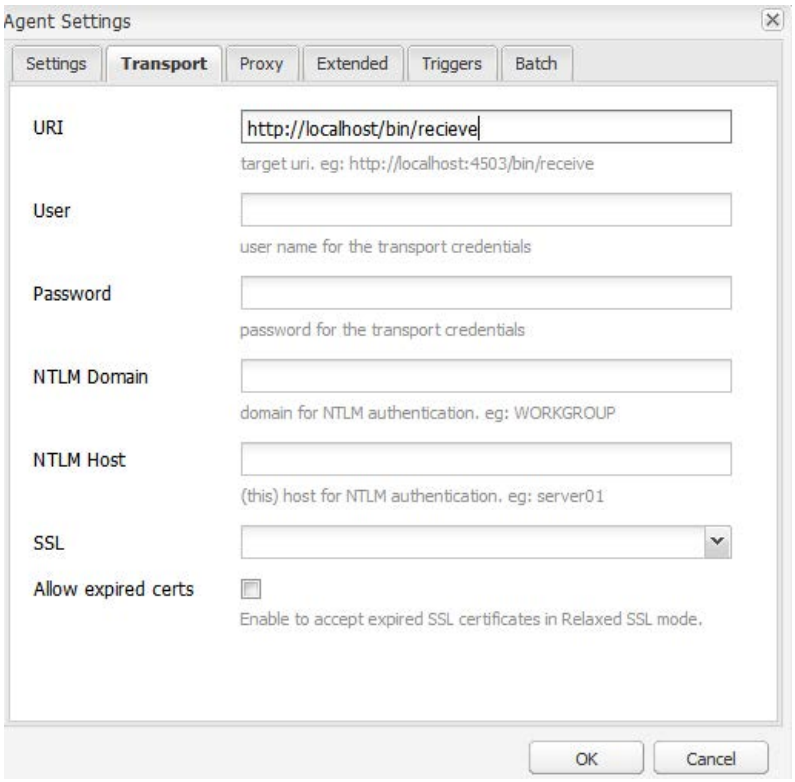
1. Open the **AEM SiteAdmin** console and click the **Tools** tab.
2. Open the required Replication Agent, for example, the **Dispatcher Flush** Agent under **Agents on Publish**, which is part of a standard installation.
3. On the **Settings** tab, ensure that **Enabled** is active.



Settings Tab of the Dispatcher Flush Agent

4. Open the **Transport** tab and enter the URI needed to access the AEM Dispatcher on the host and port where the web server is running. Enter **http://{host}:port} followed by /bin/receive**, for example: **http://localhost/bin/receive**.

NOTE: You do not have to set the user and password because this agent communicates with the AEM Dispatcher and not with another AEM instance.



Transport Tab of the Dispatcher Flush Agent

5. Replicate the **Dispatcher Flush** Agent to the Publish Instance by activating it. Use the Classic UI to perform this step.

Testing the Dispatcher Flush Agent

6. On your Author instance, publish your http://localhost:4502/editor.html/content/geometrixx/en/products/circle.html page.
7. Make sure the activation of new content was successful by accessing the page from the Publish Instance, for example: http://localhost:4503/content/geometrixx/en/products/circle.html
8. Access the page from the web server, for example, http://localhost/content/geometrixx/en/products/circle.html
9. Check the document root for a **.stat** file. The .stat file is a hidden file; so, change the folder options in Windows and Finder settings on Mac OS.
10. As a general example, it usually works like this:
    a. The CRX structure may be something like /content/yoursite/en/yourpage.
    b. The request comes in to the Dispatcher for /en/yourpage.html.
    c. AEM will serve /en/page.html successfully.
    d. The Dispatcher then serves the cache at /en/yourpage.html.
    e. The Author server sends activation request for /content/yoursite/en/yourpage.

f. Dispatcher receives the activation request and touches the stat file at:
   i. /content
   ii. /content/yoursite
   iii. /content/yoursite/en
g. The request comes into Dispatcher looking for /en/yourpage.html.
h. The stat file at /en has never been touched and statfileslevel > 0, so /en/yourpage.html is considered valid.
i. Dispatcher returns the cached file at /en/yourpage.html.

Full Dispatcher Test

1. Please make sure you have built your cache by clicking through the geometrixx site.
2. On your Author instance:
   a. open and modify your http://localhost:4502/editor.html/content/geometrixx/en/products/circle.html page in the touch authoring UI.
   b. Publish the changes done on the page.
3. Go to your Publish instance:
   a. Observe the changes done from the Author instance at http://localhost:4503/content/geometrixx/en/products/circle.html
4. Open the docroot folder and observe the following:
   a. circle.html page is deleted.
   b. .stat file and timestamp under the content/geometrixx/en folder is updated.
   c. Observe another page and it's timestamp, such as content/geometrixx/en/products.html
5. Open your Dispatcher:
   a. Go to http://localhost/content/geometrixx/en/products/circle.html and observe the page being recached.
   b. Go to http://localhost/content/geometrixx/en/products.html and observe the page's timestamp being updated in the cache

Congratulations! You successfully configured the Dispatcher with the web server, cached pages in the web server document root, and accessed activated content pages from the web server.

Oak Queries and Indexing

Objectives

- Learn about Microkernels in Oak
- Learn about DataStores
- Indexing

JCR/JSR-283

Jackrabbit Oak implements the JCR spec. The most used parts of JSR-283 are implemented, but Jackrabbit Oak does not aim to be a reference implementation of JSR-283.

Goals

- Scalability
  - Big repositories
  - Distributed repository with many cluster nodes
- Improved write throughput
  - Parallel writes
- Support for many child nodes
- Support for many ACLs

Microkernels and NodeStores

Oak provides two APIs for accessing the actual storage of the content: the **Microkernel API** and the **NodeStore API**.

While the Microkernel API is optimized for remote access to the content storage, the NodeStore API can only be accessed from within the same JVM.

The Microkernel API is completely based on strings. It uses the JSOP format, which is a lightweight HTTP protocol for manipulating JSON-based object models.

Currently, Oak has two main storage concepts:

- DocumentStore
  - Is used by DocumentMK and DocumentNodeStore
  - Stores data in MongoDB or in an RDBMS supported in AEM 6.1
- SegmentStore
  - Is used by SegmentMK and SegmentNodeStore
  - Stores data as tar files in the local file system

Design Goals and Principles

For both the **Microkernel API** and the **NodeStore API**, the following design goals apply:

- Managing huge trees of nodes and properties efficiently
- Multiversion Concurrency Control (MVCC) (Writers do not interfere with readers, snapshot isolation)
- GIT/SVN-inspired DAG-based versioning model
- Highly scalable concurrent read and write operations
- Session-less API (there is no concept of sessions; an implementation does not need to track/manage session state)
- Easy to remote
- Efficient support for a large number of child nodes
- Integrated API for efficiently storing/retrieving large binaries

Data Model

The data model of the backend data has the following characteristics:

- Simple JSON-inspired data model: just nodes and properties
- Properties represented as name/value pairs
- Supported property types: string, number, Boolean, array
- A property value stored and used as an opaque, unparsed character sequence

DocumentStore

The Oak DocumentStore manages JCR content by storing each content node in a separate document.

Implementation

The DocumentStore is a concept and is not something directly implemented. Currently, there are two relevant implementations of a document based storage:

- MongoDocumentStore: Based on MongoDB; This was implemented in AEM 6.0
- RDBDocumentStore: Based on a relational database, this is supported in AEM 6.1.

The discussion in the following sections is based on the MongoDocumentStore implementation because it is currently more mature than the RDBDocumentStore. While the concepts should apply for all implementations of the DocumentStore, it is possible that some points are implemented differently in RDBDocumentStore.

Documents

Documents in a DocumentStore are usually stored as JSON. A sample JSON-representation of a document in DocumentStore is:

```
{
     „_deleted“ : {
          „r13f3875b5d1-0-1“ : „false“
     },
     „_id“ : „1:/node“,
     „_lastRev“ : {
          „r0-0-1“ : „r13f38818ab6-0-1“
     },
     „_modified“ : NumberLong(274208516),
     „_modCount“ : NumberLong(2),
     „_revisions“ : {
          „r13f3875b5d1-0-1“ : „c“,
          „r13f38818ab6-0-1“ : „c“
     },
     „prop“ : {
          „r13f38818ab6-0-1“ : „\“foo\““
     }
}
```

Document nodes have two types of fields:

- Simple fields are stored as key/value pairs. In the example above **_id, _modified** and **_modCount** are simple fields.
- Versioned fields are kept in sub-documents where the key is a revision paired with the value of this revision.

SegmentStore

SegmentStore is an Oak storage backend that stores content as various types of records within larger segments. One or more journals are used to track the latest state of the repository. These are the principles on which SegmentStore is designed:

- Immutability: The segments are immutable, and because of that, it is easy to cache frequently accessed segments. This feature simplifies backups.
- Compactness: The size optimized is in a way that it reduces I/O costs and it fits content in caches as much as possible.
- Locality: One segment usually stores related records. With this, the cache misses are avoided, for example, if a client wants to access more related nodes per session.

Binary Data Stores

In Oak, the binary data can be stored independently from content nodes, also It is recommended that binery data should be stored externally to the node store. The most important Data Store implementations are:

- MongoDB DataStore: Binary data is stored in MongoDB.
- S3 Data Store: Binary data is stored in Amazon Cloud.
- RDB Data Store: Binary data is stored in a relational database.
- File System: Legacy implementation uses the JCR2 DataStore.

Topology Footprint	1 Author 1 Publish 1 Instance	1 Author 1 Publish 2 Instances	1 Author 2 Publish 1 Data Center	2 Author 2 Publish 2 Data Centers	2 Author 4+ Publish 2 Data Centers	4 Author 4+ Publish 4 Data Centers
Use Case	Development	Low Traffic Corporate Site	Medium Traffic Corporate Site	Medium Traffic Mission Critical	High Traffic Mission Critical	Distributed High Traffic Mission Critical
MK for Author Tier	Tar	Tar	Tar	Tar	Tar	MongoDB
MK for Publish Farm	Tar	Tar	Tar	Tar	Tar	Tar
MK for Publish Cluster	-	-	Tar	MongoDB	MongoDB	MongoDB
Datastore < 5 TB	Tar	Tar	Tar	Tar/MongoDB	Tar/MongoDB	MongoDB/Tar
Datastore > 5 TB	-	-	-	Amazon S3	Amazon S3	Amazon S3

Oak Queries and Indexing

Unlike Jackrabbit 2, Oak does not index content by default. Custom indexes need to be created when necessary, much like with traditional relational databases. If there is no index for a specific query, then the whole repository will be traversed. The query will still work but will probably be very slow.

If Oak encounters a query without an index, a WARN level log message is displayed:

```
*WARN* Traversed 1000 nodes with filter Filter(query=select
    ...) consider creating an index or changing the query
```

If this is the case, an index might need to be created, or the condition of the query might need to be changed to take advantage of an existing index.

If a query reads more than 10,000 nodes in memory, then the query is cancelled with an UnsupportedOperationException, indicating that "The query read more than 10000 nodes in memory. To avoid running out of memory, processing was stopped." As a workaround, this limit can be changed using the system property "oak.queryLimitInMemory".

Query Indices are defined under the **oak:index** node.

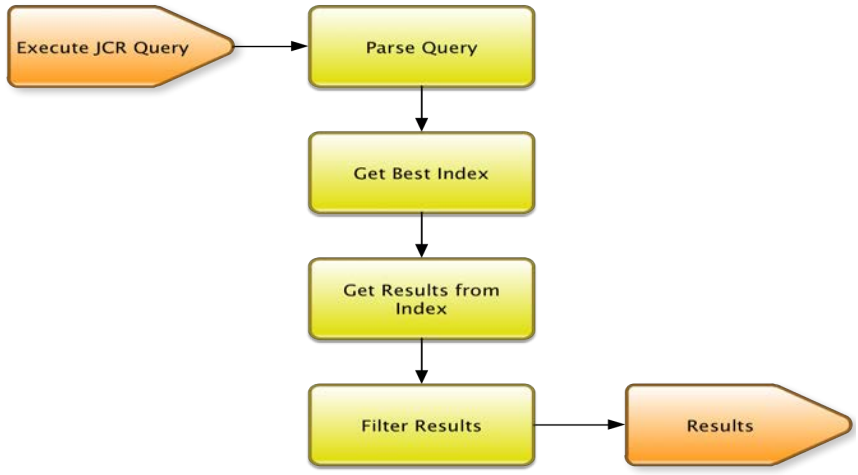
Supported Query Languages

The Oak query engine supports the following languages:

- SQL
- SQL-2
- JQOM

Oak Query Implementation Overview

The new Apache Oak based backend allows different indexers to be plugged into the repository. The standard indexer is the Property Index, for which the index definition is stored in the repository itself. External full text indexers can also be used with AEM. Implementations for Apache Lucene and Solr are available by default. The Traversal Index indexer is the one used if no other indexer is available. This means that the content is not indexed and all content nodes are traversed to find matches to the query.
If multiple indexers are available for a query, each available indexer estimates the cost of executing the query. Oak then chooses the indexer with the lowest estimated cost.



The above diagram is a high-level representation of the query execution mechanism of Apache Oak.

First, the query is parsed into an Abstract Syntax Tree. Then, the query is checked and transformed into SQL-2, which is the native language for Oak queries.

Next, each index is consulted to estimate the cost for the query. Once that is completed, the results from the cheapest index are retrieved. Finally, the results are filtered,

both to ensure that the current user has read access to the result and that the result matches the complete query.

Query Processing on cost calculations

Internally, the query engine uses a cost-based query optimizer that asks all the available query indexes for the estimated cost to process the query. It then uses the index with the lowest cost.

By default, the following indexes are available:

- A property index for each indexed property
- A full-text index, which is based on Apache Lucene/Solr
- A node type index, which is based on an property index for the properties jcr:primaryType and jcr:mixins
- A traversal index that iterates over a subtree

If no index can efficiently process the filter condition, the nodes in the repository are traversed at the given subtree.

Usually, data is read from the index and repository while traversing over the query result. There are exceptions however, where all data is read in memory when the query is executed: when using a full-text index, and when using an 'order by' clause.

Native Queries

To take advantage of features that are available in full-text index implementations such as Apache Lucene and Apache Lucene Solr, so-called native constraints are supported. Such constraints are passed directly to the full-text index. This is supported for both XPath and SQL-2. For XPath queries, the name of the function is **rep:native**, and for SQL-2, it is native. The first parameter is the index type and currently supported parameter are **Solr** and **Lucene**. The second parameter is the native search query expression. For SQL-2, the selector name (if needed) is the first parameter, just before the language. If full-text implementation is not available, the queries will fail.

Indexer Types

The **Traversal Index** indexer is used if no other indexer is available. This means that the content is not indexed and all content nodes are traversed to find matches to the query.

If multiple indexers are available for a query, each available indexer estimates the cost of executing the query. Oak then chooses the indexer with the lowest estimated cost.

Configuring the Indexes

Indexes are configured as nodes in the repository under the **oak:index** node.

The type of the index node must be **oak:QueryIndexDefinition**. Several configuration options are available for each indexer as node properties. For more information, see the following configuration details for Property Index and Lucene Index.

The Property Index

The Property Index is generally useful for queries that have property constraints but are not full-text. It can be configured using the following procedure:

- Open CRXDE Lite by going to **http://localhost:4502/crx/de/index.jsp**.
- Create a new node under **oak:index**.
- Name the node **PropertyIndex**, and set the node type to **oak:QueryIndexDefinition**.
- Set the following properties for the new node:
  type: property (of type String)
  propertyNames: jcr:uuid (of type Name)
  This particular example will index the jcr:uuid property, whose job is to expose the Universally Unique Identifier (UUID) of the node it is attached to.
- Save the changes.

  The Property Index has the following configuration options:
  a. The **type** property will specify the type of index, and in this case, it must be set to **property.**
     i. The **propertyNames** property indicates the list of the properties that will be stored in the index. In case it is missing, the node name will be used as a property name reference value. In this example, the **jcr:uuid** property, whose job is to expose the UUID of its node, is added to the index.
     ii. The **unique flag**, which if set to **true**, adds a uniqueness constraint on the property index.
     iii. The **declaringNodeTypes** property allows you to specify a certain node type that the index will only apply to.
     iv. The **reindex** flag, which if set to **true**, will trigger a full content reindex.

The Lucene Full Text Index

A full text indexer based on Apache Lucene is available in AEM 6.1.

If a full-text index is configured, then all queries that have a full-text condition use the full-text index, even if other conditions are indexed, and there is a path restriction.

If no full-text index is configured, then queries with full-text conditions may not work as expected. The query engine has a basic verification in place for full-text conditions, but it does not support all features that Lucene does and it will traverse all nodes if there are no indexed constraints.

As the index is updated through an asynchronous background thread, some full-text searches will be unavailable for a small window of time until the background processes are finished.

You can configure a Lucene full-text index using the following procedure:

1. Open CRXDE Lite and create a new node under **oak:index**.
2. Name the node **LuceneIndex** and set the node type to **oak:QueryIndexDefinition**.
3. Add the following properties to the node:
   a. **type**: lucene (of type String)
   b. **async**: async (of type String)
4. Save the changes.

The Lucene Index has the following configuration options:

- The **type** property, which will specify the type of index, must be set to **lucene**.
- The **async** property, which must be set to **async**, will send the index update process to a background thread.
- The **includePropertyTypes** propertywill define what subset of property types will be included in the index.
- The **excludePropertyNames** property will define a blacklist of property names: properties that should be excluded from the index.
- The **reindex** flag, which when set to **true**, triggers a full content reindex.

Temporarily Disabling an Index

To temporarily disable an index (for example for testing), set the index type to **disabled**. Note that while the index type is not set, the index is not updated, so if you enable it again, it might not be correct. This is especially important for synchronous indexes.

Indexing Tools

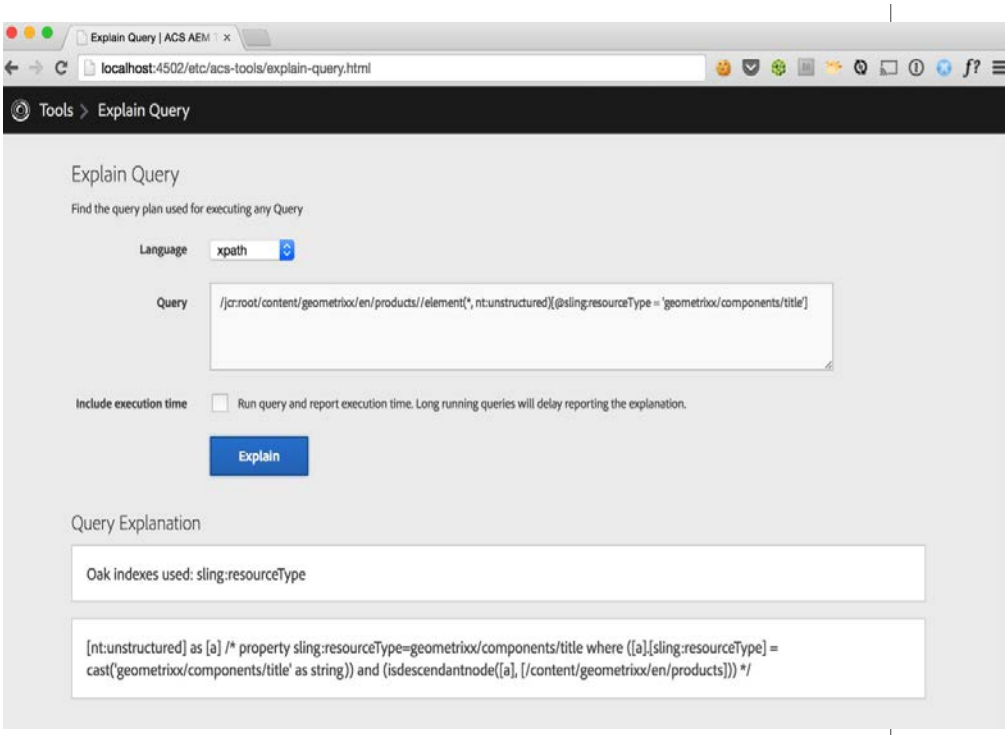
The Adobe Consulting Service Commons toolkit introduces a set of indexing tools in an effort to simplify management and maintenance of indexes in the underlining data storage layer of AEM, Apache Oak.

Explain Query Tool

This is a tool designed to help administrators understand how queries are executed. For any given query, Oak attempts to figure out the best way to execute based on the Oak indexes defined in the repository under the **oak:index** node. Depending on the query, different indexes may be chosen by Oak. Understanding how Oak is executing a query is the first step to optimizing the query.

Features

- Supports the Xpath, JCR-SQL, and JCR-SQL2 query languages
- Reports the actual execution time of the provided query
- Detects slow queries and warns about queries that could be potentially slow
- Reports the Oak index used to execute the query
- Displays the actual Oak Query engine explanation
- Provides click-to-load list of Slow and Popular queries

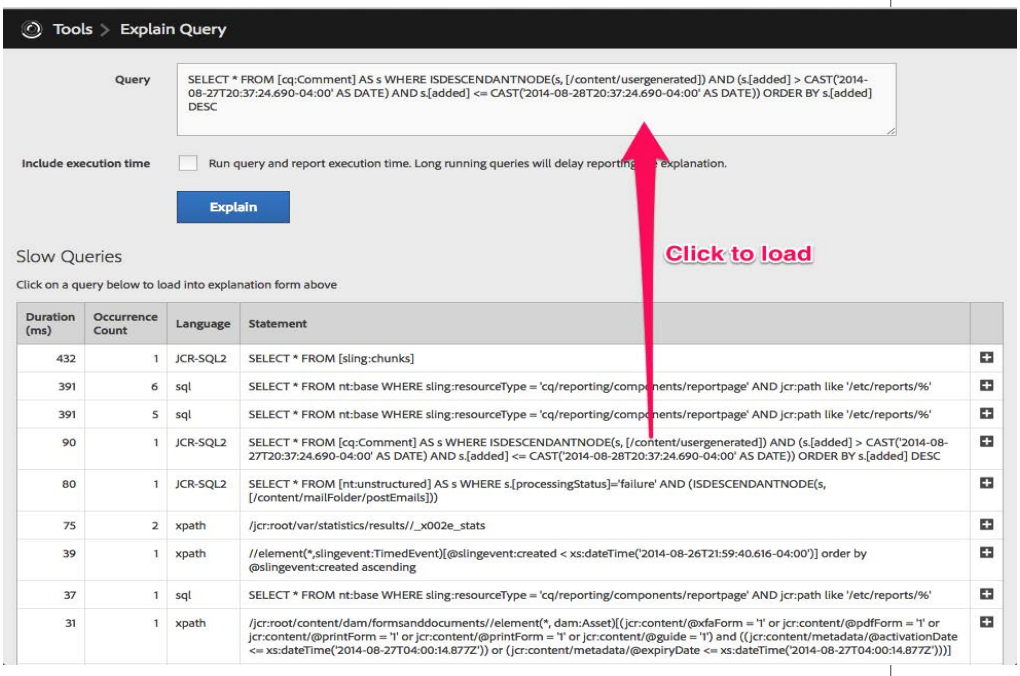


The first entry in the Query Explanation section is the actual explanation. The explanation will show the type of index that was used to execute the query.

The second entry is the query plan.

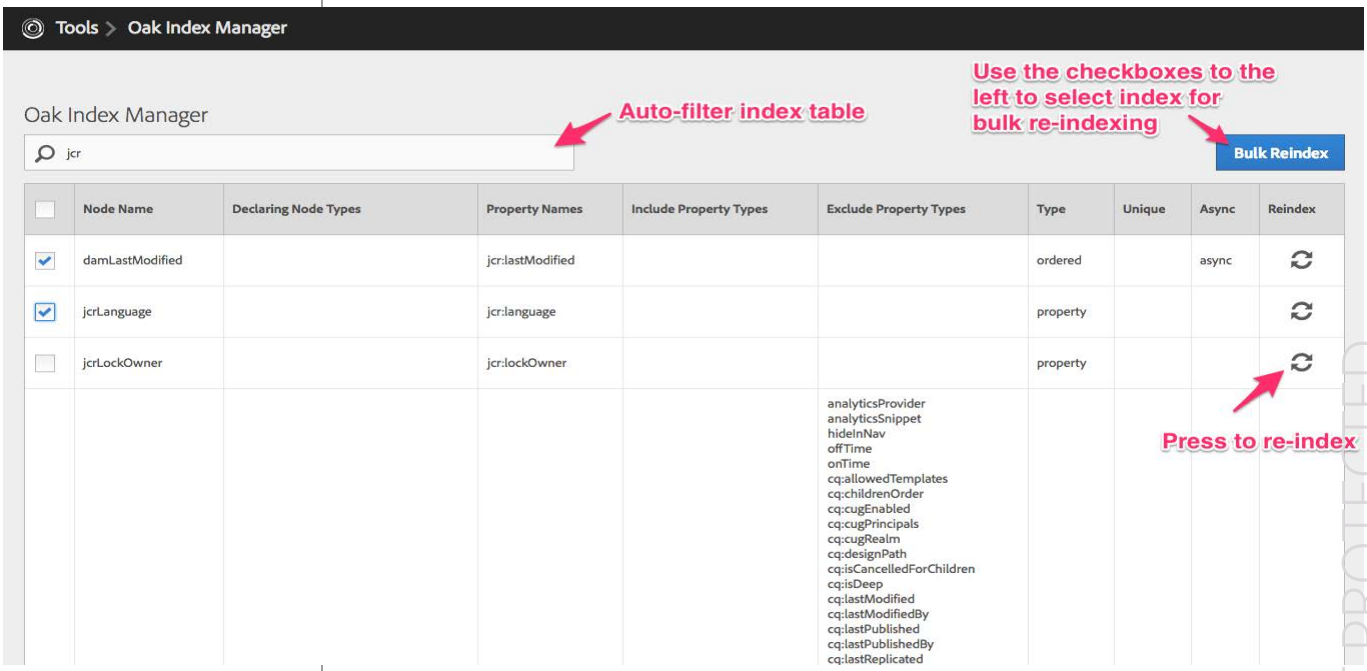
Ticking the Include exectuion time box before running the query will also show the amount of time the query was executed in, allowing for more information that can be used for optimizing the indexes for your application or deployment.

The tool will also build a list of popular and slow queries that can be automatically loaded in the UI by clicking their name in the predefined list, as shown below.



Oak Index Manager

The Oak Index Manager is a simple Web UI created to facilitate index management such as maintaining indexes, viewing their status, or triggering re-indexes.



The UI can be used to filter indexes in the table by typing in the filter criteria in the search box in the upper left corner of the screen.

A single reindex can be triggered by clicking the icon in the **Reindex** column for the respective index. A bulk reindex can be triggered by selecting multiple indexes and clicking the **Bulk Reindex** button.

With AEM 6.1, both the Explain Query and Oak Index Manager tools will be available OOTB. You can access them by going to **Tools -> Operations -> Dashboard -> Diagnosis** from the AEM Welcome screen.

Introduction to MongoDB and Relational Database MK for Oak

Objectives

- Describe the basic operations in MongoDB
- Describe document models in MongoDB
- Describe sharding concepts that MongoDB is using
- Describe replica sets in MongoDB
- Describe indexes in MongoDB
- RDBMS Support in AEM 6.1

Exercise 12.1
Install AEM With MongoDB

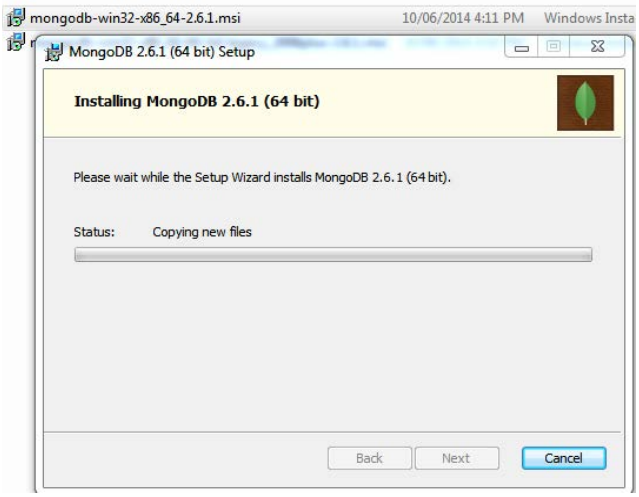
The following instructions explain how to install MongoDB and start the AEM server running on it. This exercise is divided into five sub-sections (12.1.1 to 12.1.5).
To successfully complete and understand these instructions, you will need:

- An AEM quickstart JAR file
- A valid AEM license key (in a file named license.properties)
- JDK 1.7 (this should be installed first if not already present)
- Approximately 10 GB of free disk space
- Approximately 4 to 8 GB of RAM

12.1.1: Install MongoDB

For Windows

1. Download MongoDB from http://www.mongodb.org/downloads or open it from the USB contents at <USB >distribution\MongoDB setup.
2. Follow the instructions of the installer and add the MongoDB bin directory to your path.



Set Up the MongoDB Environment on Windows

3. MongoDB requires a **data directory** to store all the data. MongoDB's default data directory path is \data\db. Create this folder using the following commands from a **command prompt**:
   `md \data\db`
4. You can specify an alternate path for data files using the --dbpath option to mongod.exe, for example:
   `C:\mongodb\bin\mongod.exe --dbpath d:\test\mongodb\data`

5. If your path includes spaces, enclose the entire path in double quotes, for example:
   `C:\mongodb\bin\mongod.exe --dbpath "d:\test\mongo db data"`

Start MongoDB

To start MongoDB, run mongod.exe. For example, from the command prompt:
`C:\Program Files\MongoDB\bin\mongod.exe`
This starts the main MongoDB database process. The Waiting for Connections message in the console output indicates that the mongod.exe process is running successfully.

For Mac OS: Use Homebrew

1. Open Terminal.
2. Update the homebrew database by running the following command:
   `brew update`
3. Install the MongoDB binaries:
   `brew install mongodb`

If you prefer to install MongoDB manually, follow the documentation at http://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/ .

12.1.2: Prepare AEM 6.1 installation for a Single MongoDB

1. Create an instance directory **aem1**.
2. Place the **aem-6.0-generic.jar** and **license.properties** files there.
3. Create a folder named **crx-quickstart**.
4. Create a folder named **install**.
5. Create a file in this folder named:
   `org.apache.jackrabbit.oak.plugins.document.DocumentNodeStoreService.cfg`
   The sample folder structure is provided at **<USB>\exercises\12.1 - aem-with_mongodb\aem1** with the following cfg file content inside **install** folder:

```
mongouri="mongodb://<host _ name _ of _ mongo _ db/ IP address>:27017"
db="OakAuthor"
service.ranking=I"50"
```

6. Create another folder structure as:

- **mongodb1**
  - **/conf**
  - **/db**
  - **/logs**

7. Create a file named **mongo.conf** in this folder with the following content:

```
dbpath=db

logpath=logs/mongo.log

logappend=true

#bind _ ip=<YourHostname/IPAddress>

port = 27017

#replSet=aem
```

A sample folder structure is provided at **<USB>\exercises\12.1 - aem-with_mongodb\mongodb1** and the configuration file is available under **\conf** directory

After the structure is build and config files are created, start mongo first and then start AEM in a seperate terminal/command prompt.

NOTE: You can choose your own name for the property 'replSet', but it must be the same for all instances of your Replica Set. Note that at this stage, replSet and bind_ip are commented out.

In this case, the MongoDB data directory is located in the AEM instance folder. You can locate it anywhere you like; just adjust the dbpath accordingly.

HINT: Always use a real host name or IP address for the parameter bind_ip. The values localhost or 127.0.0.1 will cause troubles later in a Replica Set configuration. You should always use the IP address, especially when you have multiple network adapters in your setup.
You can use ipconfig/ifconfig commands and getting the IP address of the machine and use that for configuring the bind_ip in mongo.conf and the mongouri in DocumentNodeStoreService.cfg

MongoDB will send the value configured in bind_ip to the other cluster nodes. If this value is not reachable or resolvable on the other nodes, your Replica Set will fail.

12.1.3: Start the Local MongoDB instance

On Mac OS/Unix, execute:

```
$ ulimit -n 2048
```

Change the directory to the instance folder and start MongoDB with the following command:

```
$ cd <your mongo db location>
$ mongod --config  <path _ to _ mongo _ conf>/mongo.conf
```

12.1.4: Start the AEM 6.1 instance With MongoMK

1. Open a new shell and change the current directory to the instance folder and start the AEM installation with the following command:

```
$ cd <your aem instance location>
$ java -jar -XX:MaxPermSize=256M -Xmx1024M aem-6.0.0-generic.jar -r author,crx3,crx3mongo -p 4502
```

2. Afterward, check that there is no **crx-quickstart/repository** folder (this would mean that you have installed TarMK).
3. Wait for the AEM installation to complete. Wait for browser startup. Meanwhile, you can check the log files **mongo.log** and **error.log**.

12.1.5: Verify Installation

To check for a successful MongoDB installation, execute the following steps:

1. Start a Mongo shell with the following command:
   `mongo <host>:<port>`
   Where the host value will be **localhost** in standalone setup and the port value will be **27017**, so the command will be:
   ```
   $ mongo localhost:27017
   ```
2. In the shell, execute the commands:
   ```
   $ use OakAuthor
   $ db.stats()
   ```
   which should show a non-empty database **OakAuthor**.

What Are mongo Shell, mongod, and mongos for MongoDB?

mongo is an interactive JavaScript shell interface to MongoDB, which provides a powerful interface for systems administrators and a way for developers to test queries and operations directly with the database. mongo also provides a fully functional JavaScript environment for use with a MongoDB

mongod is the primary daemon process for the MongoDB system. It handles data requests, manages data format, and performs background management operations.

mongos for 'MongoDB Shard' is a routing service for MongoDB shard configurations that processes queries from the application layer and determines the location of this data in the sharded cluster in order to complete these operations. From the perspective of the application, a mongos instance behaves identically to any other MongoDB instance.

MongoDB Microkernel (MongoDocumentStore)

The MongoDocumentStore implementation of the DocumentStore interface stores the content nodes and optionally also the binary data in MongoDB.

Oak creates the following collections for a repository in MongoDB:

```
blobs
clusterNodes
nodes
system.indexes
```

- **nodes** contain the actual repository data.
- **blobs** contains binary data (optionally, as mentioned before, binaries can be stored in a different storage than the node data).
- **clusterNodes** contains meta information about the repository.
- **system.indexes** is the default collection from MongoDB and not related to Oak.

Relational Database MK for Oak

AEM 6.1 Supports relational database persistence and it is implemented using the Document Microkernel. The Document Microkernel is the basis that is also used for implementing MongoDB persistence.

It consists of a Java API that is based on the Mongo Java API. An implementation of a BlobStore API is also provided. By default, blobs are stored in the database.

Supported Databases

The following databases are fully supported in AEM 6.1:

- DB2 10.5
- Oracle 12c

Apart from the above 2 databases AEM 6.1 also support MySQL, MariaDB and Microsoft SQL server only on experimental support. Experimental support means that you can use these databases but they are not fully tested and supported on AEM 6.1
In future release more databases will be added in the supported group.

How to configure RDBMS with AEM 6.1

The repository is created by configuring the **DocumentNodeStoreService** OSGi service. It has been extended to support relational database persistence in addition to MongoDB.

In order for it to work, a data source needs to be configured with AEM. This is done via the **org.apache.sling.datasource.DataSourceFactory.config** file. The JDBC drivers for the respective database need to be provided separately as OSGi bundles inside the local configuration.

For steps on creating OSGi bundles for JDBC drivers, please see:
https://sling.apache.org/documentation/bundles/datasource-providers.html#convert-driver-jars-to-bundle on the Apache Sling website.

NOTE: For detailed information on the data source configuration for each supported database, please refer **Data Source Configuration Options** at http://docs.adobe.com/docs/en/aem/prerelease/overview/rdbms-support-in-aem.html

Recommended Deployments and Scaling AEM With MongoDB

Recommended Deployments

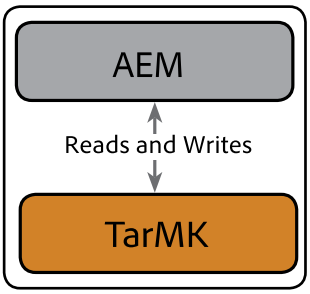
Microkernels act as persistence managers in AEM 6.1, choosing one to fit your needs depend on the purpose of your instance and the deployment type you are considering.

The following examples are meant to be an indication of their recommended uses in the most common AEM setups.

Deployment Scenarios

Single TarMK Instance

In this scenario, a single TarMK instance runs on a single server.
This is the default deployment for Author instances.



Advantages:

- Simple
- Easy maintenance
- Good performance

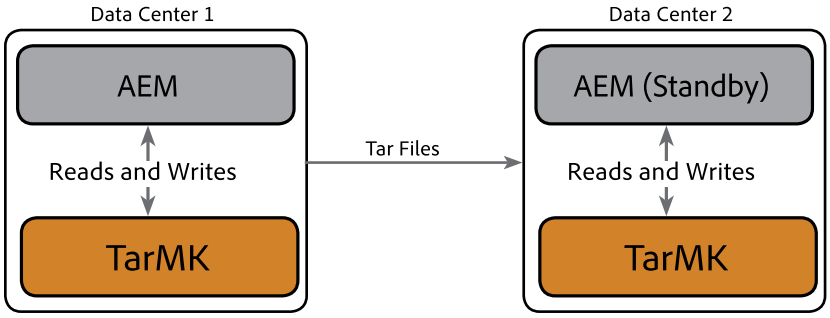
Disadvantages:

- Not scalable beyond the limits of server capacity
- No failover capacity

TarMK Cold Standby

One TarMK instance acts as the primary instance. The repository from the primary is replicated to a standby failover system.

The cold standby mechanism can also be used as a backup because the complete repository is constantly replicated to the failover server. The failover server is running in cold standby mode, which means that only the HttpReceiver of the instance is running.



Advantages:

- Simplicity
- Maintainability
- Good Performance
- Failover

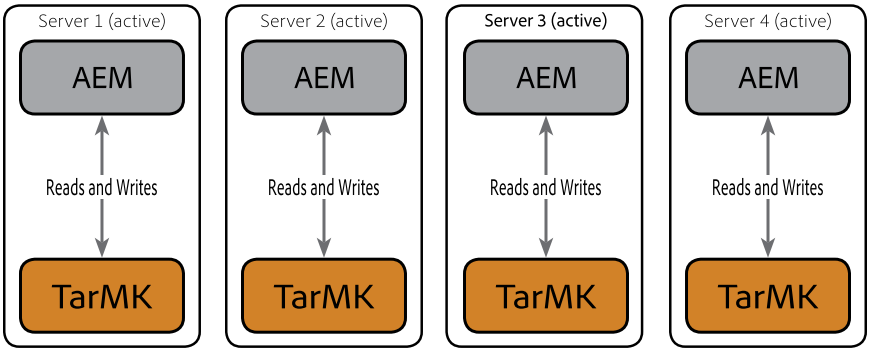
Disadvantages:

- Not scalable beyond the limits of the server capacity
- One server idle most of the time
- Failover is not automatic (It has to be detected externally before the failover system can start serving requests.)

TarMK Farm

In TarMK Farm, multiple Oak instances each run with one TarMK instance.
The TarMK repositories are independent and need to be kept in sync.

Keeping the repositories in sync is provided with the fact that the author server is publishing the same content to each farm member. User-generated content needs to be reverse replicated to the author and then published to all publish instances.



Advantages:

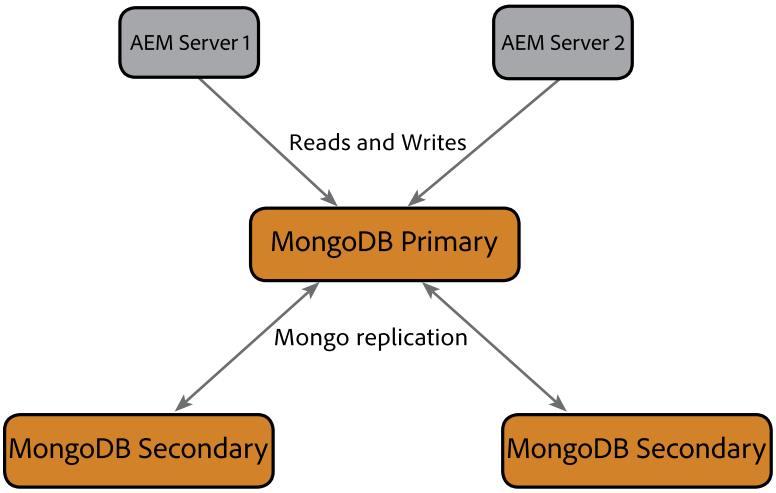
- Good Performance
- Scalability for read access
- Failover

Disadvantage:

- Write access must be kept in sync between instances

Oak Cluster with MongoMK Failover for High Availability in a Single Data center

This approach implies multiple Oak instances accessing a MongoDB replica set within a single data center. Replica sets in MongoDB are used to provide high availability and redundancy in the event of a hardware or network failure.



Advantages:

- Ability to scale AEM instances
- High availability, redundancy, and automated failover of data layer

Disadvantages:

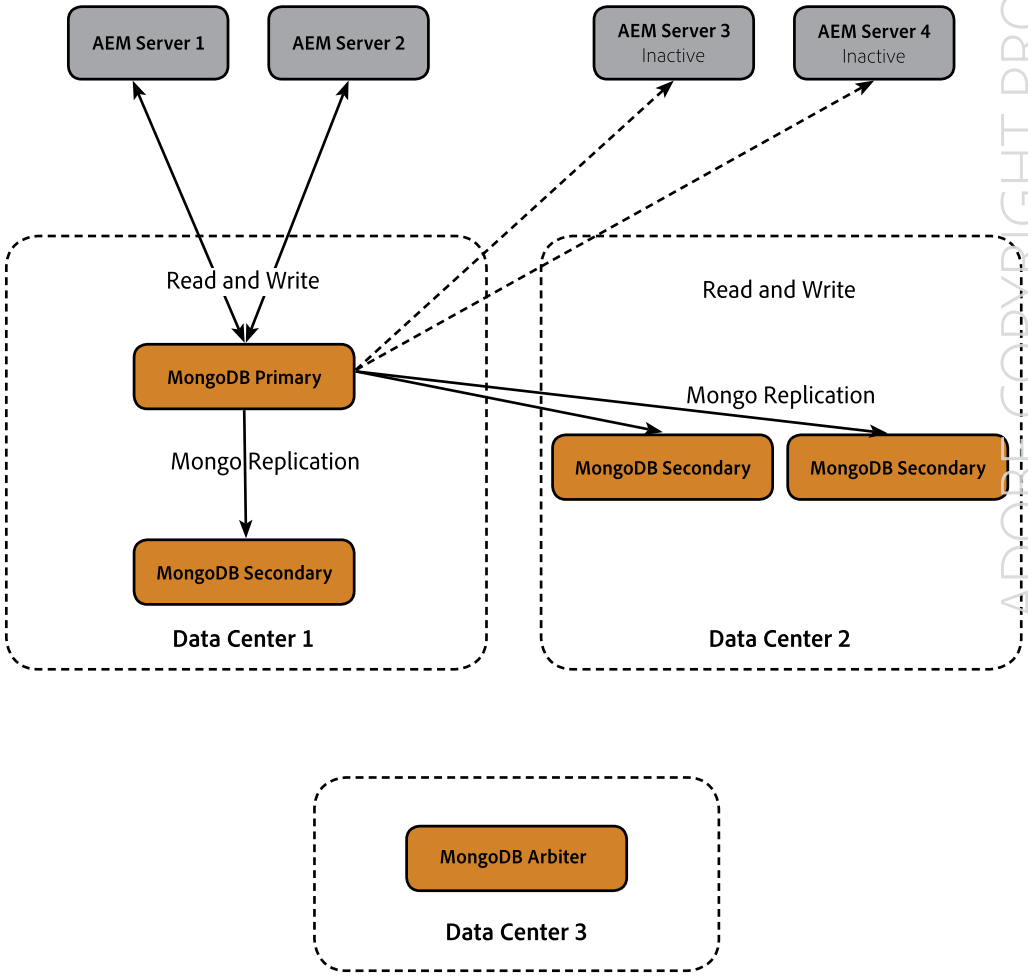
- Performance is not as good as with TarMK

Oak Cluster with MongoMK Failover Across Multiple Data centers

AEM 6.1 can leverage MongoDB persistence in order to employ full automatic failover in a multiple data center, geographically redundant deployment topology.

The failover is entirely handled by MongoDB and is independent of AEM. In order for the failover to be fully automatic, the topology needs to be deployed over at least three distinct data centers.
More specifically:

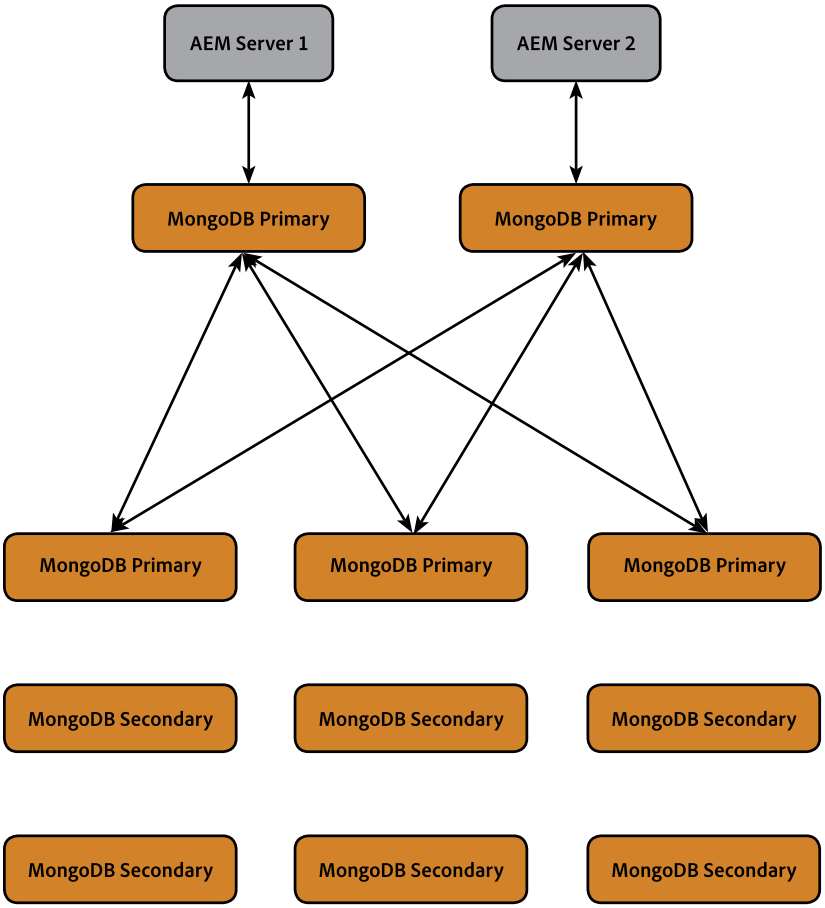
- Data center 1 is for hosting the primary MongoDB database.
- The second data center is for hosting the replica set that will provide failover in case the primary becomes unavailable.
- The third data center is for hosting the arbiter that will participate in the election process.



The databases in the first and second data centers will perform mutual backups. If the first data center suffers from an outage for any reason, one of the secondories in the second data center will automatically take over as the primary.
When the first data center is restored, the Mongo instance it hosts will detect that is no longer the primary and will join the set as a secondary.

Oak Cluster With Sharded MongoDB

Multiple Oak instances access a sharded MongoDB deployment. MongoDB provides a built-in capability for scaling horizontally called Sharding. This is typically used for data sets that scale beyond the capacity of a single replica set. Sharded deployments of MongoDB are built to leverage the same high availability and redundancy characteristics provided by a replica set while also supporting linear scaling of reads and writes.



Advantage:

- Ability to scale AEM instances
- Horizontal scalability of data layer
- Supports geographically distributed AEM deployments

Disadvantage:

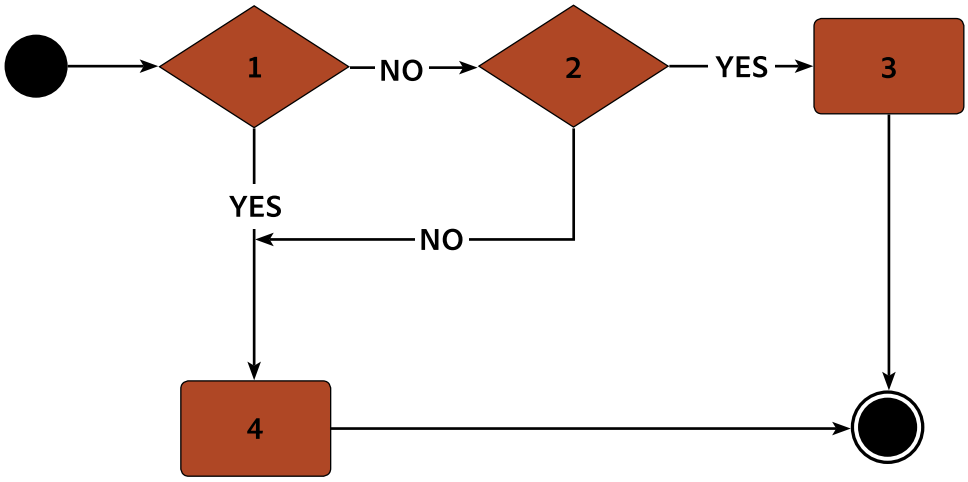
- Complex to operate

Microkernels: Which One to Use

The basic rule that needs to be taken into account when choosing between the two available Microkernels is that TarMK is designed for performance, while MongoMK is used for scalability.

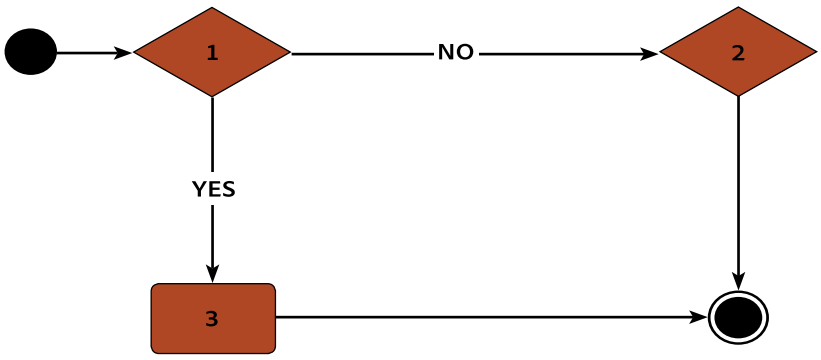
To establish the best type of deployment suited to your requirements, you can use following decision matrices.

Choosing the Deployment Type for Author Instances



1	Are you able to meet your requirements regarding repository size, server CPU, and write throughput on one server?
2	Is it possible to extend the server capacity to meet your requirements?
3	Use an Oak cluster with Mongo secondaries.
4	Use a single TarMK instance.

Choosing the Deployment Type for Publish Instances



1	Does your site have user-generated content?
2	Use a TarMK farm.
3	Use TarMK with cloud storage or MongoMK Active/Active cluster depending on the volume and type of the user-generated content.

Choosing the deployment type for Social Communities

For sites which plan to incorporate social communities features, a certain deployment is required due to user-generated content (UGC) resulting from site visitors using the features to post content.

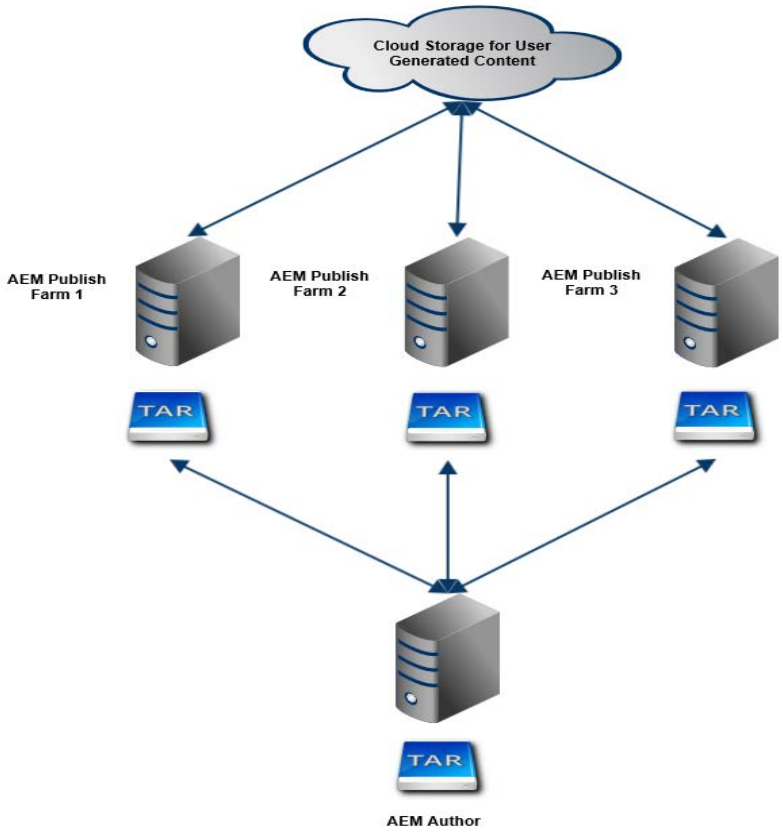
UGC is posted from the published site, but administration occurs in the author environment and moderation occurs in both author and publish environments. Thus, there is a need to replicate the content and moderation in both directions in order to obtain a consistent view of the UGC.

An option to storing the UGC on the author and publish environments is to store the UGC in a common store on Adobe Social cloud, which requires a special license.

TarMK with Cloud Storage

Based on the Tar micro kernel for persistence management, it uses a separate cloud storage for all the user-generated content.

This deployment is recommended for sites with high volumes of user-generated content (millions of posts per year).



Advantages:

- The cloud storage option is configured as a cloud service in AEM 6.1 and can be activated on each page individually
- Can scale to very large number of posts
- Moderation of user-generated content takes place in the cloud

The disadvantages:

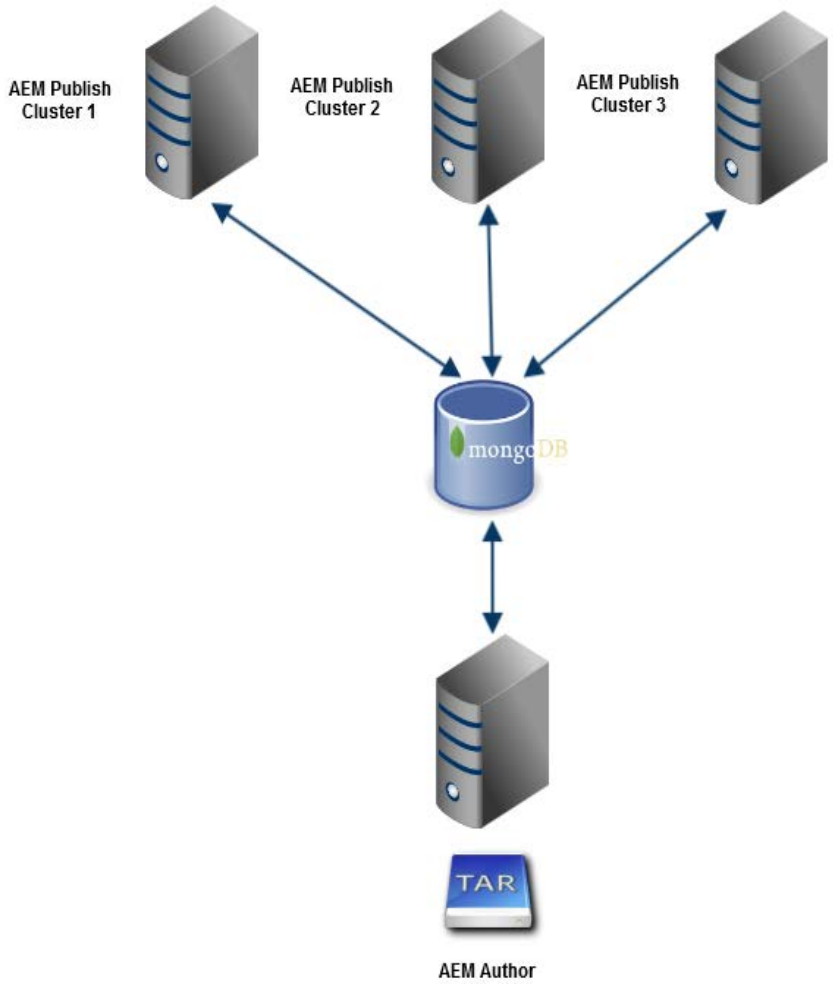
- Doesn't completely remove the need for reverse/forward replication. Some pages created on the publish side (like activity streams) will still need to be replicated on the author side.

MongoMK with Active-Active Cluster

Based on the Mongo microkernel, all the user-generated content is centrally stored inside a Mongo instance.

This deployment is recommended for sites that have a high business impact and normal user-generated content volumes. This generally includes enterprise community platforms and B2B websites.

Also note that this deployment is currently tied to Social Communities. Using it for other types of purposes needs upfront analysis.



Advantages:

- Horizontal scalability
- No replication needed, even for the content created on the publish side
- Moderation is done on the publish side

NOTE: MongoDB is third-party software and is not included in the AEM licensing package. For more information, see the MongoDB licensing policy page. In order to get the most of your AEM deployment, Adobe recommends licensing the MongoDB Enterprise version in order to benefit from professional support.
The license includes a standard replica set, which is composed of one primary and two secondary instances that can be used for either the author or the publish deployments.
In case you wish to run both author and publish on MongoDB, two separate licenses need to be purchased.

Exercise 13.1
Setup Mongo Clusters

1. First, stop all instances of AEM 6.1 and MongoDB (Ctrl+C), which were created in previous exercises.
2. Create a Mongo structure for db2 and db3. Folder structure is included in **<USB>\exercises\13.1 - mongodb_cluster_setup**
   - mongodb2 (same structure as mongodb1)
   - mongodb3 (same structure as mongodb1)
3. Modify all the **mongo.config** files for your mongo databases. Folder structure for the same is available at:
   - **<USB>\exercises\13.1 - mongodb_cluster_setup\mongodb1\conf**
   - **<USB>\exercises\13.1 - mongodb_cluster_setup\mongodb2\conf**
   - **<USB>\exercises\13.1 - mongodb_cluster_setup\mongodb3\conf**

```
dbpath=db

logpath=logs/mongo.log

logappend=true

#bind _ ip=<YourHostname/IPAddress>

port = 27017

#replSet=aem
```

   a. Each mongo.conf should point to a different data directory (**dbpath**) where the instance stores its data and contain the same lines as shown earlier.
   b. Uncomment the **#replSet=aem** field in all the mongo.conf files (remove the #), and these must all point to the same replication set. Also, uncomment the **#bind_ip** line.

c. Also, remember to have a different port in each config file (that is, 27107, 27108, 27109).

These instances can be on the same or on different machines. If they are on different machines, check if the new instances are reachable from the first instance, and check that all instances can resolve the host name configured in the value of 'bind_ip' of all other instances.

4. Start all the MongoDB instances.
   ulimit is only applicable for Mac machines:
   ```
   ulimit -n 2048
   mongod --config crx-quickstart/conf/mongo.conf
   ```

5. When all MongoDB instances are running, connect the mongo shell to the initial MongoDB instance (which already contains the AEM 6.1 repository):
   ```
   mongo <host_name_of_mongo_db>:27017/OakAuthor
   ```

6. Then execute the following commands:
   Initialize the Replica Set
   ```
   rs.initiate()
   ```
   Show the Replica Set config. It should show only one member
   ```
   rs.conf()
   ```

7. Add all the other MongoDB nodes to the Replica Set (NOTE: The quotes are literal):
   ```
   rs.add("<secondhost>:27018")
   rs.add("<thirdhost>:27019")
   ```

8. Check Replica Set configuration.
   ```
   rs.conf()
   rs.status()
   ```

9. Validate the cluster configuration. The commands above should show the initial instance plus the instances you just added. The initial instance should be in the state PRIMARY, all other instances in the state STARTUP2 or SECONDARY. See the online MongoDB documentation for a description of these states.

CAUTION: Execute the `rs.initiate()` commands only on the initial MongoDB!

Exercise 13.2
Configure AEM 6.1 Instances With MongoDB Cluster

We will configure the initial AEM 6.1 instance to use the MongoDB cluster and additional AEM instances.

1. Start all the AEM instances you want to use.

2. The folder structure for the three AEM instances are provided at:
   - **<USB>\exercises\13.2 - config_aem_with_mongodb-cluster\author1**
   - **<USB>\exercises\13.2 - config_aem_with_mongodb-cluster\author2**
   - **<USB>\exercises\13.2 - config_aem_with_mongodb-cluster\author3**

Place the Quickstart jar and license file under **author1\crx-quickstart**, **author2\crx-quickstart** and **author3\crx-quickstart** folders. If you are running all instances on the same machine, use port numbers 4502, 4504 and, 4506 for the instances.

Avoid 4503 as the default out-of-the-box Replication Agents may cause unexpected results.

3. Configure the following in all the instances under **\crx-quickstart\install** directory:
   `org.apache.jackrabbit.oak.plugins.document DocumentNodeStoreService`
4. Update the property 'mongouri' appropriately to reflect your hosts and their respective ports.
   `mongodb://<host1>:<port>,<host2>:<port>,<hostN>:<port>`
   Here, you list all the MongoDB instances in your Replica Set; for example:
   `mongodb://<YourHostname/IPAddress>:27017,<YourHostname/IPAddress>:27018,<YourHostname/IPAddress>:27019`

   
   NOTE: Use full machine name rather than localhost or 127.0.0.1.
5. Start **author1, author2,**and **author3** so that they use the cfg file parameters.
6. See **rs.status()** for a list of all the instances.

Congratulations! You just set up your first MongoDB cluster with AEM 6.1.

Exercise 13.3
Validate the AEM/MongoDB Cluster

We will test that the changes performed in one AEM instance reflect in the other instances:

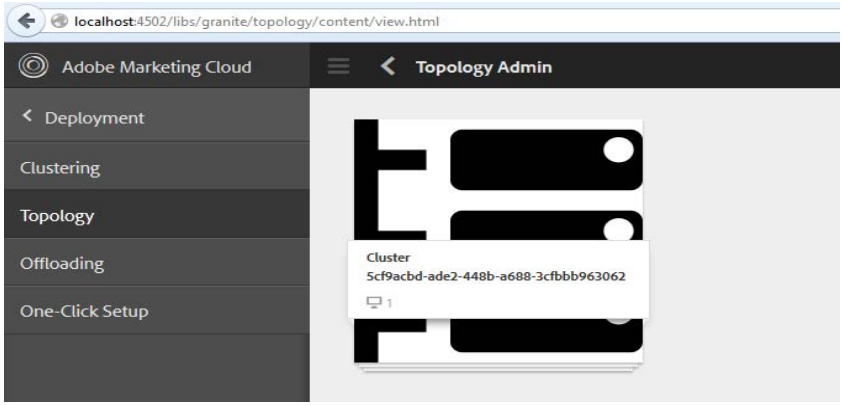
1. In one of the AEM instances, create a page named **Clustered**.
2. Open a new browser window, log in to another AEM instances, and check that the page is also appearing in that instance.

Test the High Availability of the Cluster

1. Stop the first MongoDB instance you installed.
2. Check how you can still access any of the AEM instances.
   a. Remember that you will need at least two running MongoDB instances running at any time for the cluster to work.
   b. Use **rs.status()** on the mongo console connected to a running MongoDB instance to see the master-slave relationships.

Managing CRX3 Clusters

You can manage your clusters by using **Topology Admin** by navigating to **Tools -> Operations -> Deployment -> Topology**. You can directly navigate to http://localhost:4502/libs/granite/topology/content/view.html for the same.

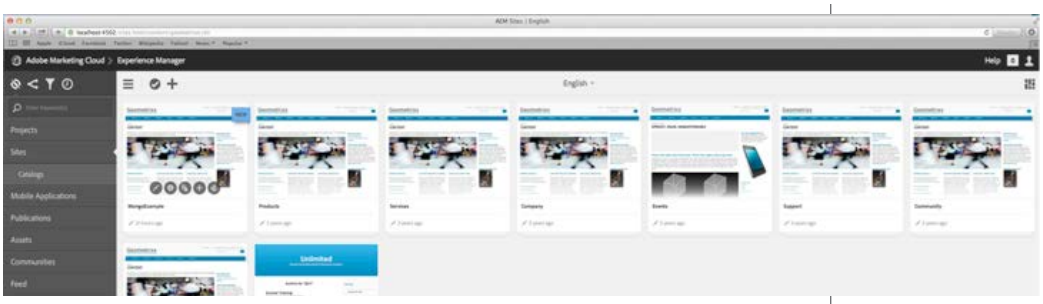


After selecting the cluster, you can view the list of instances running. Cluster view will give you information about the current and leader instances for your cluster.

Exercise 13.4
Explore AEM 6.1 with MongoDB

In exercise 12.1, we have started an AEM instance, which is using MongoDB. Now, we will explore how MongoDB is maintaining the repository.

1. AEM 6.1 expects a running MongoDB instance. At first startup, AEM 6.1 currently expects the MongoDB to run on **localhost** and on the default port **28017**. You should already have a mongod instance running from the previous exercises; if not, start it now.
2. Start the AEM 6.1 instance with the following command. Notice the –r parameter with sets the sling runmode to crx3 and crx3mongo.
   ```
   java -jar cq-quickstart-6.0.0-***.jar -r crx3,crx3mongo
   ```
3. Your browser should open the AEM Welcome screen; this might take a few minutes. Log in with **admin/admin** Credentials.
4. Click **Sites**, then **Geometrixx Demo Site**, and finally **English**.
5. Create a new page with the title **MongoExample**, and then add a child page to MongoExample with the title **MongoPage1**.



6. Now, try to find this page in your mongo database. In the terminal/command line, start the mongo process if not already started from a previous exercise.

7. Execute the mongo command to change the current working database to Oak:

```
use Oak
```

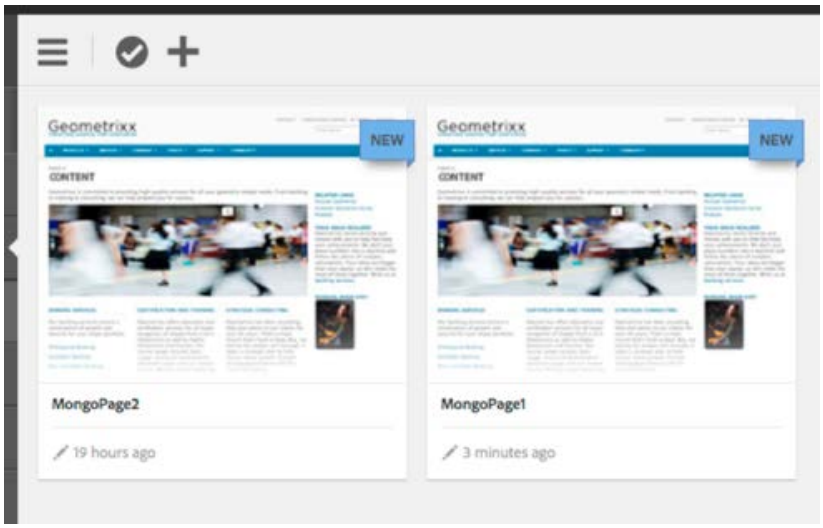
8. Try to find the page, you have created. Execute the following command:

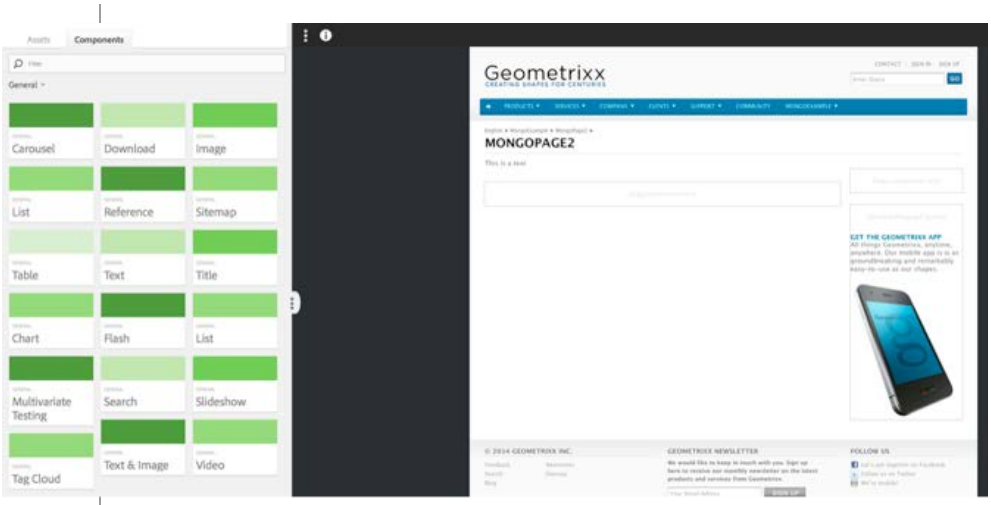
```
db.nodes.find({ _id:{$regex:'[0-9]:/content.*/mongoexample.*'}},{ _id:1})
```

9. You should receive the following result:

```
{ "_id" : "4:/content/geometrixx/en/mongoexample" }
{ "_id" : "5:/content/geometrixx/en/mongoexample/jcr:content" }
{ "_id" : "5:/content/geometrixx/en/mongoexample/mongopage1" }
{ "_id" : "6:/content/geometrixx/en/mongoexample/mongopage1/jcr:content" }
```

10. Switch to the AEM 6.1 browser and create one more page under MongoExample named **MongoPage2**. Open this page and add a text component to it.





11. In the mongo shell, execute the same command again:
    Now, you will also receive the component in your console. You will receive a result that looks something like this:

```
db.nodes.find({ _id:{$regex:'[0-9]:/content.*/mongoexample.*'}},{ _id:1}).
```

```
{ " _id" : "4:/content/geometrixx/en/mongoexample" }
{ " _id" : "5:/content/geometrixx/en/mongoexample/jcr:content" }
{ " _id" : "5:/content/geometrixx/en/mongoexample/mongopage1" }
{ " _id" : "5:/content/geometrixx/en/mongoexample/mongopage2" }
{ " _id" : "6:/content/geometrixx/en/mongoexample/mongopage1/jcr:content" }
{ " _id" : "6:/content/geometrixx/en/mongoexample/mongopage2/jcr:content" }
{ " _id" : "7:/content/geometrixx/en/mongoexample/mongopage2/jcr:content/par" }
{ " _id" : "8:/content/geometrixx/en/mongoexample/mongopage2/jcr:content/par/text" }
```

12. Go to your browser again, delete **MongoPage1**, and see what happens in the database.

13. Execute this again:

```
db.nodes.find({ _id:{$regex:'[0-9]:/content.*/mongoexample.*'}},{ _id:1})
```

    You notice that you will receive the same result again. That means, this node will continue to exist in the mongo database.

14. Find out how AEM 6.1 knows that the page has been deleted. Execute the following command in the mongo shell:

```
db.nodes.find({ _id:{$regex:'[0-9]:/content.*/mongoexample/mongopage1'}})
```

    In the result, you should see the deleted property. Notice how Oak stores the page that was deleted only for a specific revision:

```
":childOrder": {
    "r1449b9ee0c5-0-2": "[\"nam:jcr:content\"]"
}, "_children": true, "_commitRoot": {
    "r1449b9ee0c5-0-2": "0",
    "r1449ba748e3-0-2": "0"
}, "_deleted": {
    "r1449b9ee0c5-0-2": "false",
    "r1449ba748e3-0-2": "true"
}, "_id": "5:/content/geometrixx/en/mongoexample/mongopage1", "_
    lastRev": {
    "r0-0-2": "r1449ba748e3-0-2"
}, "_modCount": NumberLong(3), "_modified" :
    NumberLong(278836167), "jcr:created" : { "r1449b9ee0c5-0-2" :
    "\"dat:2014-03-07T09:18:04.603+01:00\"" }, "jcr:createdBy" :
    { "r1449b9ee0c5-0-2" : "\"admin\"" }, "jcr:primaryType" : {
    "r1449b9ee0c5-0-2" : "\"nam:cq:Page\"" } }
{ ":childOrder" : … }
```

Backing Up AEM 6.1

Purpose of Backup

The purpose of backup is not to provide a failover as we are already running cluster for mongo and Oak to support that. The purpose of the backup is:

- Disaster Recovery
- Data Recovery in case of data corruption
- Adding a new node to cluster (by recreating the node from backup)

TarMK Backup

For backup purposes, the whole AEM instance can be copied and zipped into the local file system.

Backups can be triggered through Java Management Extensions (JMX) console OR curl. In essence, this is identical to an earlier AEM backup method.

MongoMK Backup

Use MongoDB tools to create a backup of the repository database.

Preferred sequence:

- Create a replica of the MongoDB primary.
- Optional: Create a checkpoint on MongoDB to handle non-merged conflicts.
- Stop replication.
- Perform a backup.
- Restart replication.

Optionally, Mongo Service MMS (mms.mongodb.com) can be used to create a backup in the cloud.

- You have to use the Mongo shell to run the following commands to backup/restore the aem-author database. Note that you need to provide the valid values for <localhost/IP address>, <db name>, and <path to save backup> as per your actual implementation.

  mongodump —host <localhost/IP address> —port 27017 —db <db name> —out <path to save backup>

- Restore the aem-author database from a backup, after dropping the existing aem-author database.

  mongorestore -host <localhost/IP Address> -port <port> <path to backup>

DataStore Backup (Binaries)

In general, it is recommended to run the DataStore garbage collection before creating a backup.

There are two ways to back up and restore CRX repository content:

1. You can create an external backup of the repository and store it in a safe location. If the repository breaks down, you can restore it to the previous state.
2. You can create internal versions of the repository content. These versions are stored in the repository along with the content, so you can quickly restore nodes and trees you have changed or deleted.

General

The approach described here applies for system backup and recovery.

If you need to back up and/or recover a small amount of content that is lost, a recovery of the system is not necessarily required:

- Either you can fetch the data from another system through a package, or
- You can restore the backup on a temporary system, create a content package, and deploy it on the system where this content is missing.

Timing

Do not run backup in parallel with the DataStore garbage collection, as it might harm the results of both processes.

Generally, it is recommended to run the Tar Optimizer after the backup has been performed.

Offline Backup

You can always perform an offline backup. This requires a downtime of CRX, but can be quite efficient in terms of time required compared to an online backup.

In most cases, you will use a file system snapshot to create a read-only copy of the storage at that time. To create an offline backup, perform these steps:

- Stop the application.
- Make a snapshot backup.
- Start the application.

Online Backup

This backup method creates a backup of the entire repository, including any applications deployed under it, such as AEM. The backup includes content, version history, configuration, software, hotfixes, custom applications, log files, search indexes, and so on. If you are using clustering and the shared folder is a subdirectory of crx-quickstart (either physically or using a softlink), the shared directory is also backed up.

You can restore the entire repository (and any applications) at a later point.

This method operates as a hot or online backup and it can be performed while the repository is running. Therefore, the repository is usable while the backup is running. This method works for the default, TarPM-based CRX instances.

When creating a backup, you have the following options:

- Backing up to a directory using AEM's integrated backup tool.
- Backing up to a directory using a filesystem snapshot

In any case, the backup creates an image (or snapshot) of the repository. Then, the system's backup agent should take care to actually transfer this image to a dedicated backup system (tape drive).

Database

The online backup only backs up the file system. If you store the repository content and/or the repository files in a database, that database needs to backed up separately.

CRX Online Backup

An online backup of your repository lets you create, download, and delete backup files. It is a 'hot' or 'online' backup feature, so it can be executed while the repository is being used normally.

When starting a backup, you can specify a **Target Path** and/or a **Delay.**

Target Path

The backup files are usually saved in the parent folder of the folder holding the quick-start jar file (.jar). For example, if you have the CRX jar file located under */InstallationKits/CRX*, then the backup will be generated under */InstallationKits*. You can also specify a target to a location of your choice.

If the **TargetPath** is a directory, the image of the repository is created in this directory. If the same directory is used multiple times (or always) for storing backup:

- modified files in the repository are modified accordingly in the TargetPath
- deleted files in the repository are deleted in the TargetPath
- created files in the repository are created in the TargetPath

If TargetPath is set to filename with the extension .zip, the repository is backed up to a temporary directory, and then the content of this temporary directory is compressed and stored in the ZIP file.
This approach is discouraged, because:

- It requires additional disk storage during the backup process (temporary directory plus the zip file)

- The compression process is done by the CRX and might influence its performance.
- It delays the backup process.
- Up to Java 1.6, Java was only able to create ZIP files up to a size of 4 gigabytes. Java 1.7 onwards you can create more then 4 GB zip files during backup.

If you need to create a ZIP as the backup format, you should backup to a directory and then use a compression program to create the zip file.

Delay

Indicates a time delay (in milliseconds), so that repository performance is not affected.

By default, the CRX backup runs at full speed. You can slow down creating an online backup, so that it does not slow down other tasks.

A delay of one millisecond typically results in 10% reduction in CPU usage, and a delay of 10 milliseconds usually results in less than 3% CPU usage. The total delay in seconds can be estimated as follows:

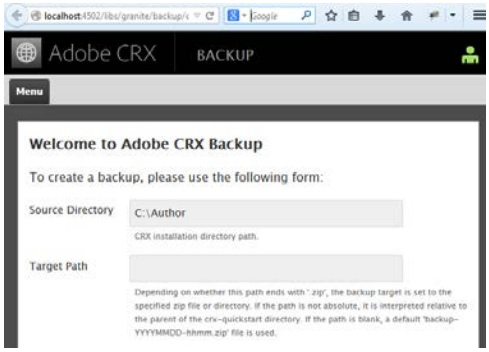
- Repository size in MB * delay in milliseconds / 2 (if the zip option is used)
- Repository size in MB * delay in milliseconds / 4 (when backing up to a directory)

That means a backup to a directory of a 200 MB repository with 1 millisecond delay increases the backup time by about 50 seconds.

You can access backups (made to both zip files and target directories) from the file system. From here, you can use any kind of backup tool (full backup or incremental backup).

Exercise 14.1
Create a Backup

1. Log in to AEM as the administrator (admin) and choose: **Tools-> Operations-> Backup** in the Touch UI or open http://localhost:4502/libs/granite/backup/content/admin.html directly from your browser.



2. The backup console will open. Specify the **Target Path** and **Delay** (delayed to a later time) as required:
3. Click **Start**; a progress bar will indicate the progress of the backup. You can cancel a running backup at any time.
4. When the backup is complete, the zip files are listed in the left pane of the console.

Congratulations! You successfully created a full backup of your Author repository without taking the instance down.

NOTE: Backup files that are no longer needed can be removed using the console. Select the backup file in the left pane, and then click Delete. If you have backed up to a directory, after the backup process is finished, the CRX will not write to the target directory.

Backing Up the DataStore Separately

If the file DataStore has been configured outside the main repository, it is not included in the backup. This will reduce the size of the online backup and the size of the backup zip file. However, the DataStore also needs to be backed up. As files in the DataStore directory are immutable, they can be backed up incrementally (potentially using sync) or after running the online backup.

NOTE: Do not run the DataStore backup and garbage collection concurrently.

Backing Up to the Default Target Directory

CAUTION: In the following example, various parameters in the curl command might need to be configured for your instance; for example, the host name (localhost), port (4502), admin password (xyz), and file name (backup.zip).

```
curl -u admin:admin -X POST http://localhost:4502/system/console/jmx/com.adobe.
    granite:type=Repository/op/startBackup/java.lang.String?target=backup.zip
```

The backup file/directory is created on the server in the parent folder of the folder containing the crx-quickstart folder (the same as if you were creating the backup using the browser). For example, if you have installed CRX in the directory /InstallationKits/crx-quickstart/, then the backup is created in the /InstallationKits directory.

The curl command returns immediately and therefore you must monitor this directory to see when the zip file is ready. While the backup is being created, a temporary directory (with the name based on that of the final zip file) can be seen—at the end, this will be zipped.

For example:

- Name of the resulting zip file: backup.zip
- Name of the temporary directory: backup.f4d5.temp

Backing Up to a Non-default Target Directory

Usually, the backup file/directory is created on the server in the parent folder of the folder containing the crx-quickstart folder. If you want to save your backup (of either sort) to a different location, you can set an absolute path to the target parameter in the curl command.

For example, to generate backupJune.zip in the directory /Backups/2012:

```
curl -u admin:admin -X POST http://localhost:4502/system/console/jmx/com.adobe.
    granite:type=Repository/op/startBackup/java.lang.String?target=/Backups/2012/
    backupJune.zip"
```

CAUTION: When using a different application server (such as JBoss), the online backup may not work as expected because the target directory is not writable. In that case, contact Support.

Backing Up a Large Repository

As your repository grows, the size can start to affect the length of time required to make a backup. This in turn can affect either downtime or performance of the application.

There are various options available for you to consider when making backups of a large repository:

Snapshot Backup

A snapshot backup involves taking a read-only copy of a storage device at a given moment. They are designed to be instantaneous, or as close as possible.

To make a snapshot backup of your application (for example, CRX or AEM), you need to:

1. Stop the application.
2. Make a snapshot backup.
3. Start the application.

As the snapshot backup usually takes only a few seconds, the entire downtime is less than a few minutes; if you are running a cluster, you only need to stop and back up one cluster node and therefore there is no downtime.

Incremental Backups

The online backup can write to a target directory instead of a zip file. With this, you can achieve both full and incremental backups.

After the online backup is finished, you can backup this target directory using either a snapshot or a classical incremental backup tool.

CAUTION: Incremental backup of the file DataStore is supported; when using incremental backup for other components (such as Lucene index), ensure deleted files are marked as 'deleted' in the backup as well.

Online Backup with Time Delay

You can also configure a time delay when performing online backups to minimize the impact the backup has on the performance of CRX (or AEM).

Separate Data Store

Storing your data store outside the repository allows it to be backed up separately. It also allows you to take incremental backups of the data store, which will be quicker than a full backup (though sporadic full backups are recommended to provide a base point, should a restore ever be required).

The datastore directory can be backed up at runtime. DataStore garbage collection must not be run until the backup is finished.

Custom Log Files

Goal

Various AEM log files provide detailed information about the current system state. In addition to the default system log files, you can also create and customize your own log files. They can help you better track messages produced by your own applications and to separate them from the default log entries.

AEM's logging system is based on Simple Logging Facade for Java (SLF4J), consisting of two elements, a Logging Logger and a Logging Writer. The Writer persists the data provided by the Logger to a configurable location. Usually, it is a file. The Logger collects data from different components inside AEM, filters them by requested severity level, and redirects the output to a configured Writer.

In AEM, you can configure logging for the two major interfaces, AEM and CRX, independently. Usually, you will be more interested what happens under AEM's cover. In particular cases, you can also set up a higher verbosity level for the CRX module.

In this section, you will generate a new log file and monitor only messages produced by a specific set of AEM modules.
There are two ways to configure the AEM's Logging system—using the Apache Felix Web Console and CRX. Adobe recommends configuring everything in CRX and using the Apache Web Console as a viewer only. The exercise describes both ways.

As an advanced topic, you also configure logging for CRX to monitor its processes.

Exercise 6.1
Create a Custom Log File

Create a Custom Log File With a Specified Log Level

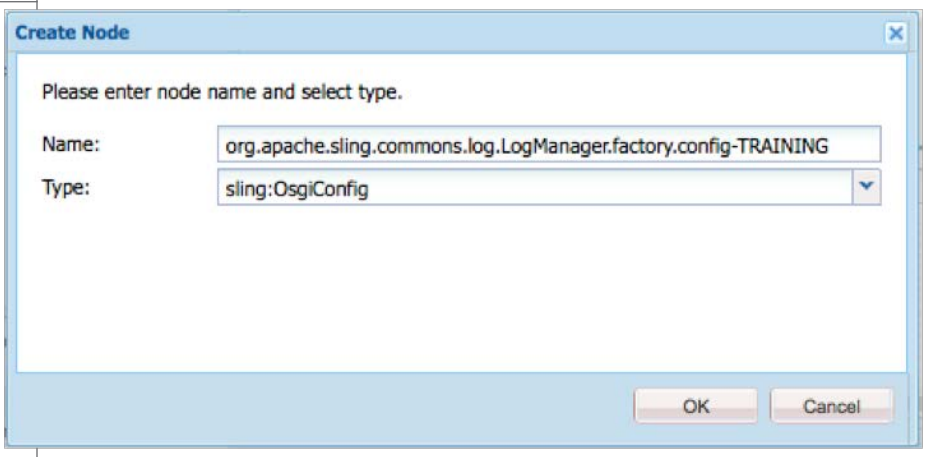
First, you create the configuration in CRX, and then you review it in the Apache Felix console. A first view of possible configuration modules in Apache (http://localhost:4502/system/console/configMgr) shows two possible configurations—single modules and factory modules. The difference is that a single module exists only once in CRX, while from a factory module, several modules can be created. As an example, the module 'Apache Sling Logging Configuration' is a singleton module, while 'Apache Sling Logging Logger Configuration' is a factory module used to create as many Logger modules as required of the same type.

Create the Logging Logger

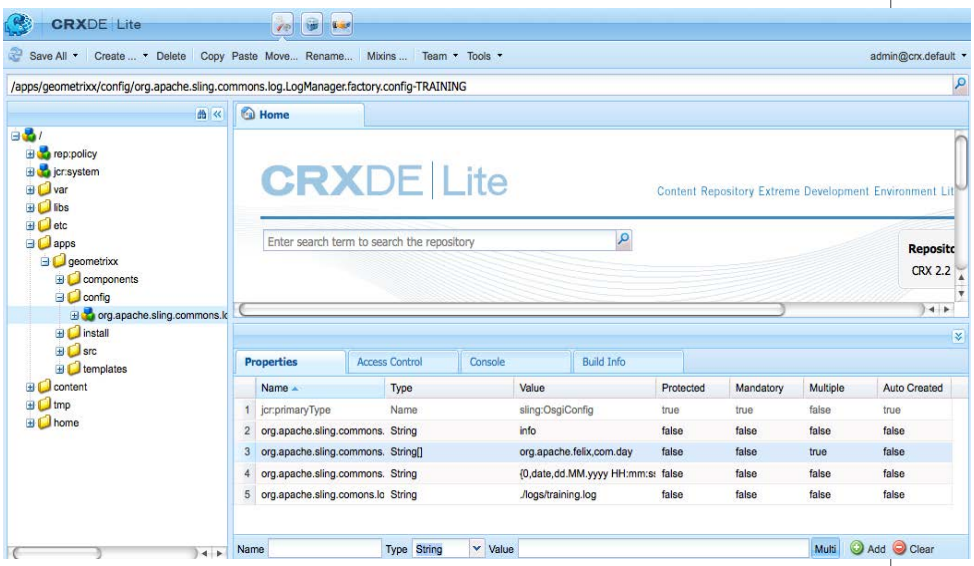
In this first example, you create a new Logger module based on a factory module (Apache Sling Logging Logger Configuration) using CRX. It should log protocol messages generated by the modules `org.apache.felix` and `com.day`, filtering out messages with log severity higher than 'Debug'.

1. Open CRXDE Lite (http://localhost:4502/crx/de/) so that you can define a new configuration for the custom log file.
2. If the **/apps/geometrixx/config.author** folder does not already exist, do the following:
   a. Right click the **/apps/geometrixx** folder and choose **Create... Create Node**:
      i. **Name:** config.author
      ii. **Type:** sling:Folder
   b. Click **Save All**.
3. Using the same steps as described in step 2, create a subnode of **/apps/geometrixx/config.author**. Use the values:
   a. Name: org.apache.sling.commons.log.LogManager.factory.config-TRAINING
   b. Type: sling:OsgiConfig

NOTE: We are creating the custom Logger in this section and Writer in the next section to explain it for existing project (Geometrixx). Best practice is that you should create the same under following:
1. /apps/<project-name>/config.* for projects specific OSGi configurations
2. /apps/system/config.* for configurations that affect that particular system, like LibraryManager, DebugFilter, and so on
3. /apps/global/config.* for configurations for all AEM instances across multiple run modes



4. Set the following properties on the newly created node:
   a. **Name:** org.apache.sling.commons.log.file
   b. **Type:** String
   c. **Value:** logs/training.log
   d. **Name:** org.apache.sling.commons.log.level
   e. **Type:** String
   f. **Value:** Info
   g. **Name:** org.apache.sling.commons.log.pattern
   h. **Type:** String
   i. **Value:** {0,date,dd.MM.yyyy HH:mm:ss.SSS} *{4}* [{2}] {3} {5}
   j. **Name:** org.apache.sling.commons.log.names
   k. **Type:** String[] (String Array) (String + Multi)
   l. **Values:**
      i. org.apache.felix
      ii. com.day



Configuration properties for the new log file

5. Do not forget to save the new properties by clicking **Save All**.

Create the Logging Writer

A logging Writer is only necessary for a configuration that is different to the default one. The default Writer will select a default size of 10 MB, and 5 as the default number of files. Limit the file size of your Logger to 1 MB per file, and keep 3 log files on the hard disk.

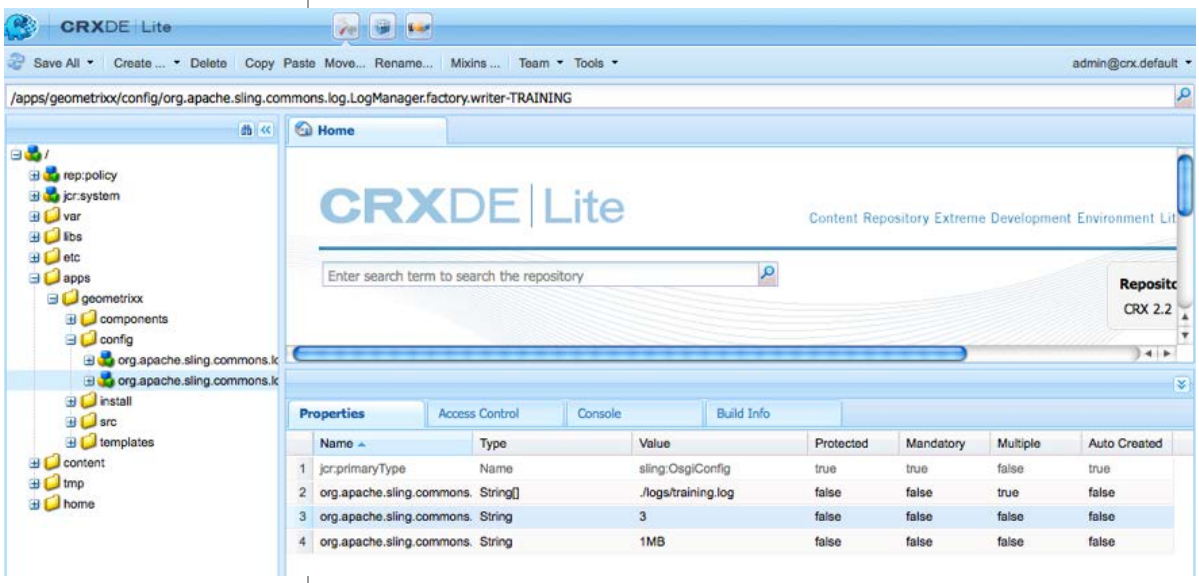
1. Under **/apps/geometrixx/config.author**, create a node for the new 'Apache Sling Logging Writer Configuration' using the following settings:
   a. **Name**: org.apache.sling.commons.log.LogManager.factory.writer-TRAINING
   b. **Type**: sling:OsgiConfig

2. Set the following properties on the newly created node:
   a. **Name:** org.apache.sling.commons.log.file
      Type: String
      Value: logs/training.log

   b. **Name:** org.apache.sling.commons.log.file.size
      Type: String
      Value: 1MB

   c. **Name:** org.apache.sling.commons.log.file.number
      Type: Long
      Value: 3

3. Click **Save All** to save the settings.



Added the Log Writer settings

4. Open some AEM pages to produce log entries and examine your log file.

Run Modes

Run modes allow you to tune your AEM instance for a specific purpose, for example, author, publish, or development. This is done by defining collections of configuration parameters for each run mode. A basic set is applied for all run modes; additional sets are each tuned to the purpose of your specific environment. AEM has built-in author or publish run modes (do not remove these). You can add additional custom run modes if required. For example, you can configure run modes for:

- **Environment:** local, dev, test, prod
- **Location:** Berlin, Basel, Timbuktu
- **Company:** acme, partner, customer
- **Special system type:** importer

Setting Run Modes

It is possible to define specific run mode(s) a specific instance should run on. By default, an author instance runs on run-mode author and a Publish Instance runs on run-mode publish. It is possible to define several run modes for one instance; for example, author, foo, and dev. These run-modes have to be set as VM options. For example, from the command line:
java -Dsling.run.modes=author,foo,dev -Xmx1024m -jar cq-quickstart-5.6.0.jar

Alternatively, in the start script:

```
# default JVM options
CQ_JVM_ OPTS="-Dsling.run.modes=author,foo,dev"
```

or by adding entries to the crx-quickstart/conf/sling.properties file, for example:

```
sling.run.modes=author,dev,berlin
```

Configurations Per Run Mode

To create separate configuration settings per run mode, create folder names in the form:

```
config.<runmode> are used, e.g. "config.publish":
/apps/myapp/config.publish
```

For systems with run-modes publish and berlin:

```
/apps/myapp/config.publish.berlin
```

Configurations for Different Run Modes

Some examples of configuration settings that may be needed for different run modes:

Different mail server configurations per location:

- config.basel/com.day.cq.mailer.DefaultMailService
- config.berlin/com.day.cq.mailer.DefaultMailService

Enabling or disabling debugging per environment:

- config.prod/com.day.cq.wcm.core.impl.WCMDebugFilter
- config.dev/com.day.cq.wcm.core.impl.WCMDebugFilter

Additional Information on Run Modes

When using different configurations for separate run modes, the following apply:

- Partial configurations are not supported.
- The configuration with maximum matching run modes wins.

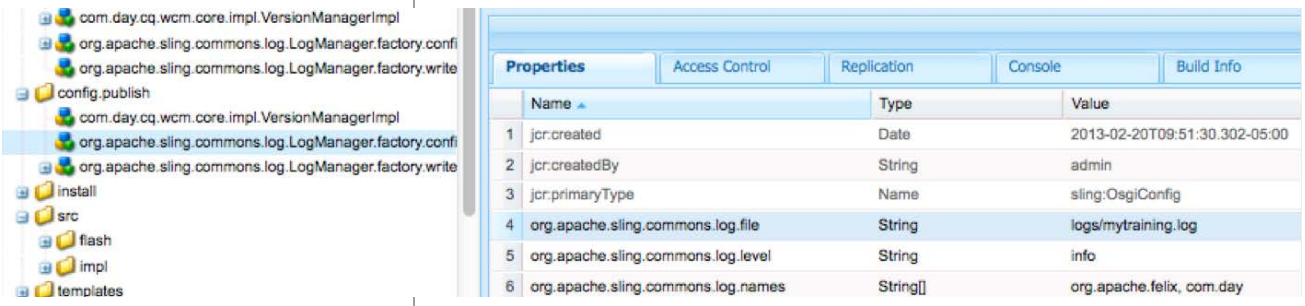
To avoid unexpected results:

- Always set all properties to avoid confusion.
- Use a type indicator (for example, {Boolean}, {String}) in every property.

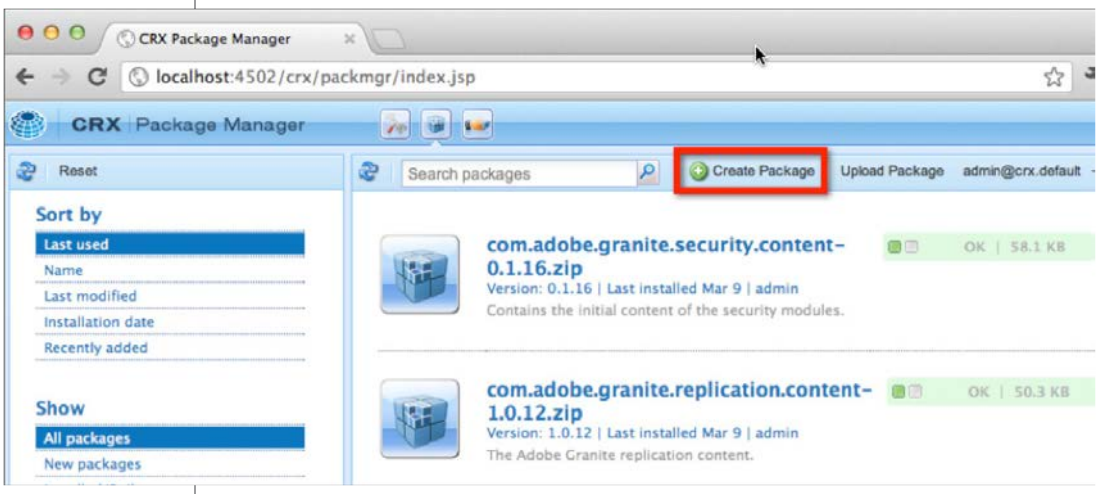
Using Custom Run Modes With Configurations

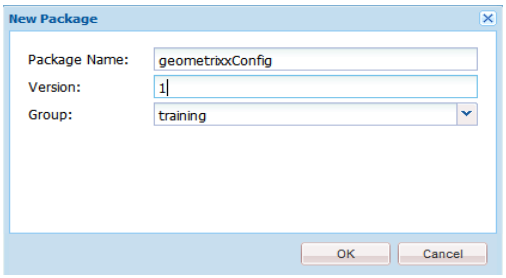
Now that you know how to configure OSGi bundles, and you have author and Publish instances, you can examine how custom run modes affect bundle configuration. In this next exercise, you will cause the training log file to have a different name on the Publish Instance as compared to the author instance.

1. Copy **/apps/geometrixx/config.author** to **/apps/geometrixx/config.publish**.
2. Modify the values of the `org.apache.sling.commons.log.file` property on `org.apache.sling.commons.log.LogManager.factory.config-TRAINING` and `org.apache.sling.commons.log.LogManager.factory.writer-TRAINING` nodes
   a. **Name**: org.apache.sling.commons.log.file
      Type: String
      Value: logs/mytraining.log

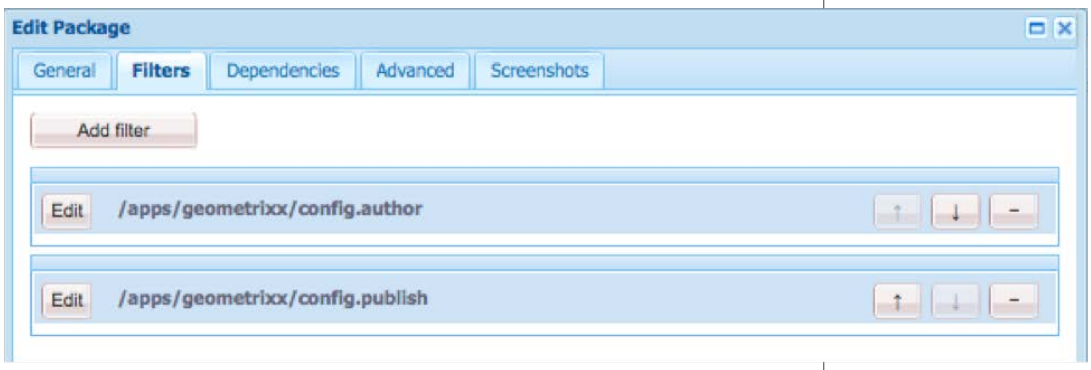


3. Open Package Manager from Tools-> Operations-> Packaging-> Packages or directly by opening http://localhost:4502/crx/packmgr/index.jsp
4. Click **Create Package**. **Name:** geometrixxConfig, **Group:** training.



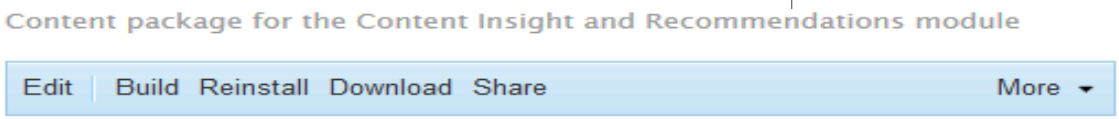


5. Define two filters: **/apps/geometrixx/config.author** and **/apps/geometrixx/config.publish**. **Save.**

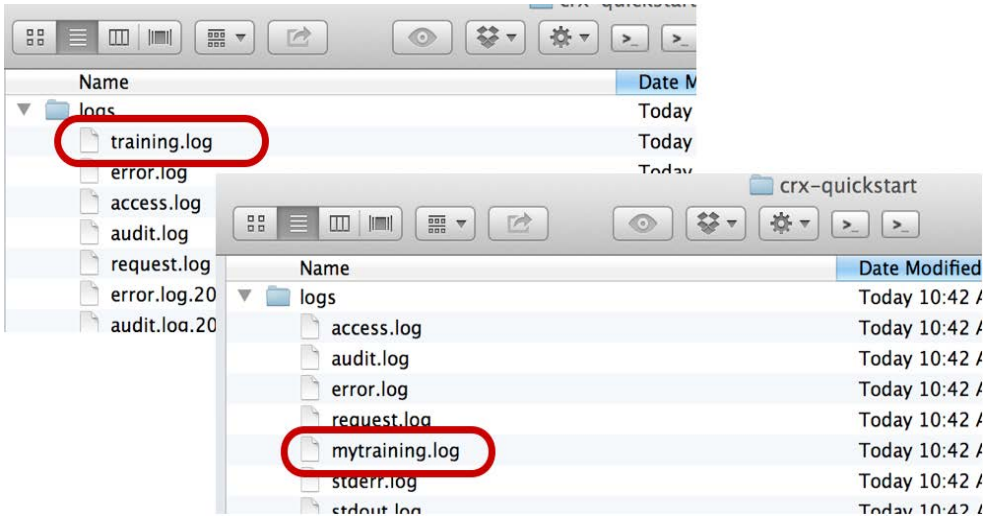


6. Build the package.

7. Navigate Package created and click on **More** and select **Replicate**.



8. Test your configuration by looking in the **crx-quickstart/logs** directory to see the **training.log** file in the author instance and the **mytraining.log** file in the Publish Instance.

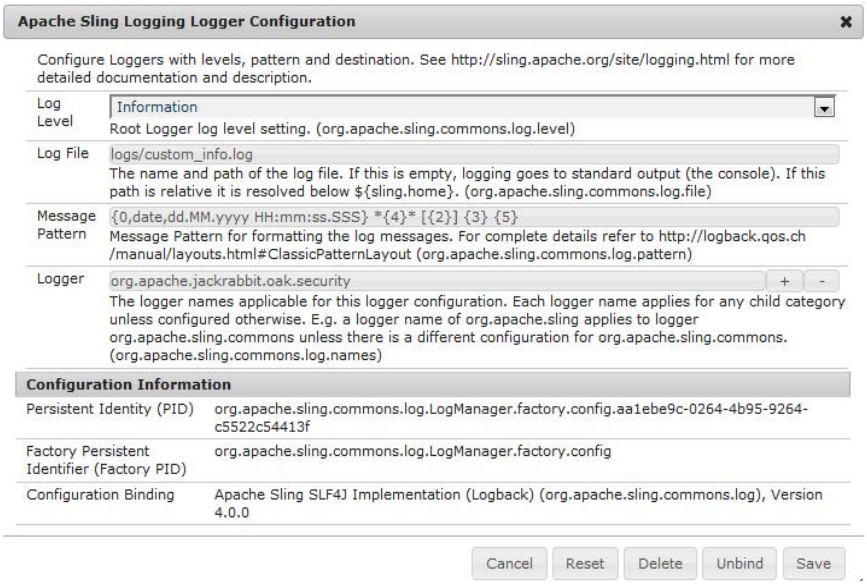


Congratulations! You deployed application configurations and seen how the instance chooses the right configuration from the instance run mode.

Create a CRX Logger

In addition to an AEM Logger, you also want to monitor what happens at the CRX level. CRX and AEM consists of different modules. In this example, you are going to create a new log file to monitor the activities performed by the indexing service of CRX, which is provided by the underlying Jackrabbit search engine

1. On your browser, navigate to http://localhost:4502/system/console/configMgr and look for 'Apache Sling Logging Logger Configuration'.
2. Click the button with a plus sign to create a new factory configuration.
3. In the section **Loggers**, add your Logger :
   a. **Log Level:** Information
   b. **Log File:** logs/custom_info.log
   c. **Message Pattern:** {0,date,dd.MM.yyyy HH:mm:ss.SSS} *{4}* [{2}] {3} {5}
   d. **Logger:** org.apache.jackrabbit.oak.security



Logger Configuration

4. Click **Save.**

After a while, consider to either comment out the additional Logger and appender or set its severity level higher (to Info, Warn, or Error); otherwise, your system is slowed down unnecessarily and a considerable amount of hard disk space is consumed.

Congratulations! You successfully created your own custom log files. You can also examine the configuration of your AEM Logger from the Apache Felix Console at the following URL: http://localhost:4502/system/console/slinglog

NOTE: Editing a Logger configuration created within CRX from the slinglog console is not recommended as it will replace the configuration nodes corresponding to the Logger with a file, although it will continue to work.

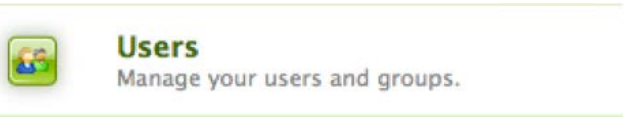
User Administration and Security

Goal

This chapter describes how to configure and manage user authentication and authorization within the AEM scope. To successfully complete and understand these instructions, you will need:

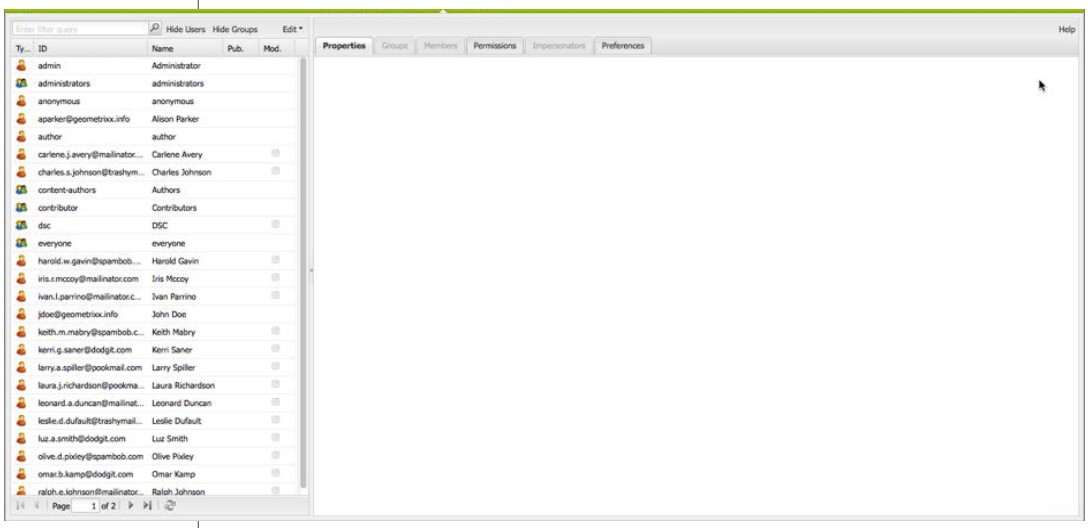
- A running AEM Author instance

Users and Groups



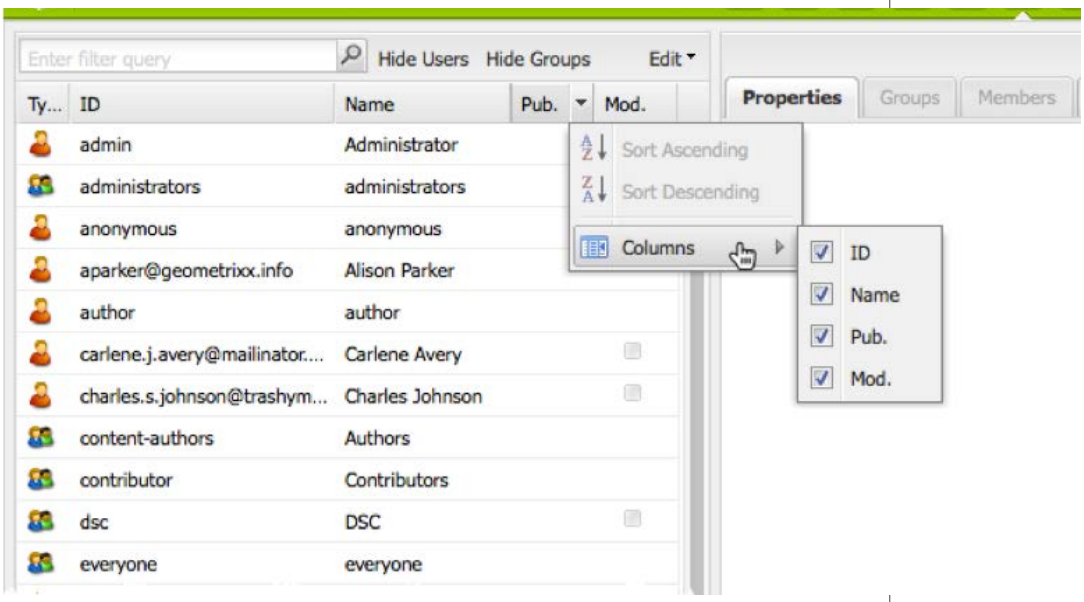
Users: A user models either a human user or an external system connected to the system. The user account holds the details needed for accessing AEM. A key purpose of an account is to provide the information for the authentication and login processes—allowing a user to log in. Each user account is unique and holds the basic account details, together with the privileges assigned. Users are often members of groups, which simplify the allocation of these permissions and/or privileges.

Groups: These are collections of users and/or other groups; these are all called members of a group. Their primary purpose is to simplify the maintenance process by reducing the number of entities to be updated, as a change made to a group is applied to all members of the group.



Security Pane Used for Managing Users and Groups

The left tree lists all the users and groups currently in the system. You can select the columns you want displayed, sort the contents of the columns, and even change the order in which the columns are displayed by dragging the column-header to a new position.



Change the Order or Select What Columns to Display

Now, you will create a group and provide some access rights and restriction—as they appear in your future projects. Later, create a user, and make that user a member of the newly created group. Finally, test if your environment reacts as expected.

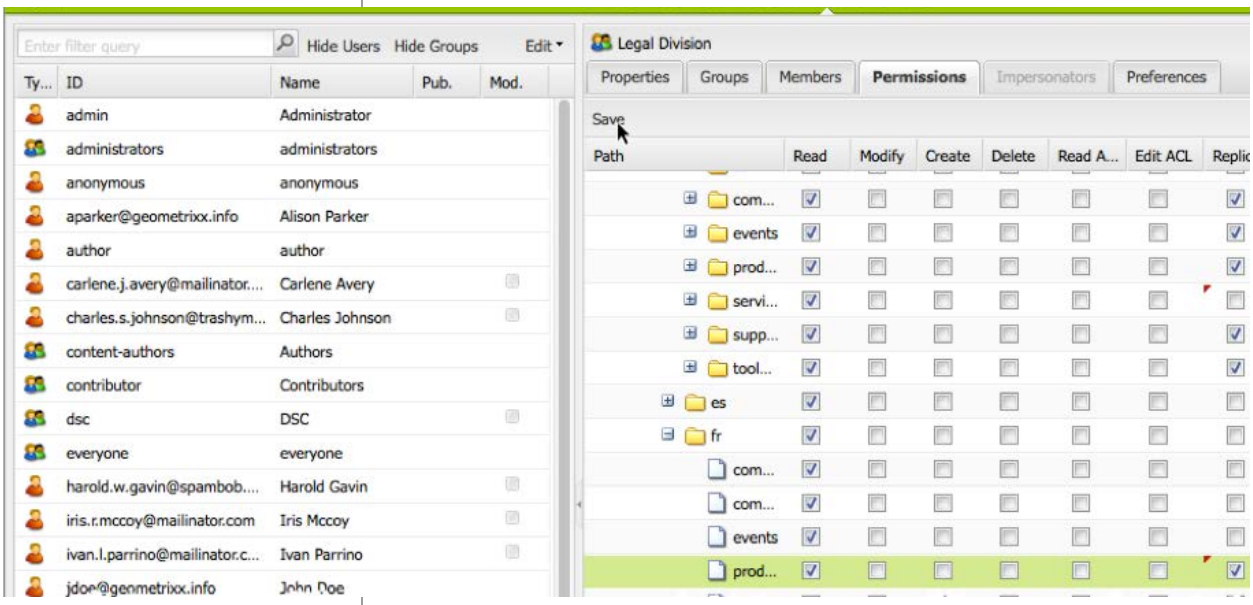
The requirement list for this new group looks like this:

- Provide access only to the console's Websites, Digital Assets, and Inbox. That means denying access to the other ones (Campaigns, Community, Tools, Users, Tagging).
- Members of this new group are allowed to modify content of already existing pages located under Geometrixx > English. Add new pages but do not delete any.
- Pages located under Geometrixx > French (Français) should be accessed in read-only mode.
- The Geometrixx > German (Deutsch) page is not accessible at all (not visible) to members of the group.
- Users are allowed to activate or deactivate (replicate) pages located under Geometrixx> English. The exception is the Services page and all pages stored below. You do not have permission to replicate pages under Geometrixx > French, except the Products page.
- Access to Design mode for Geometrixx pages is allowed.

Exercise 15.1
Create a New User

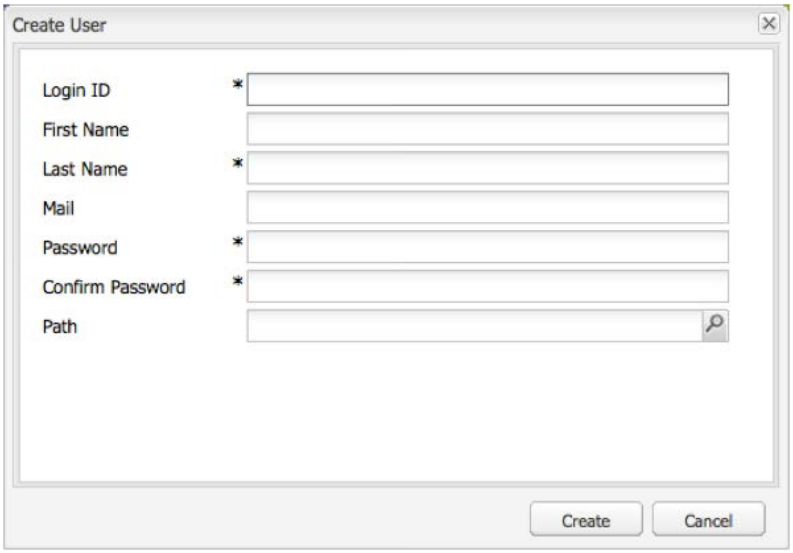
Add yourself as a user in AEM. Use the following steps:

1. In the **Security** window tree list, click **Edit** > **Create** > **Create User**.



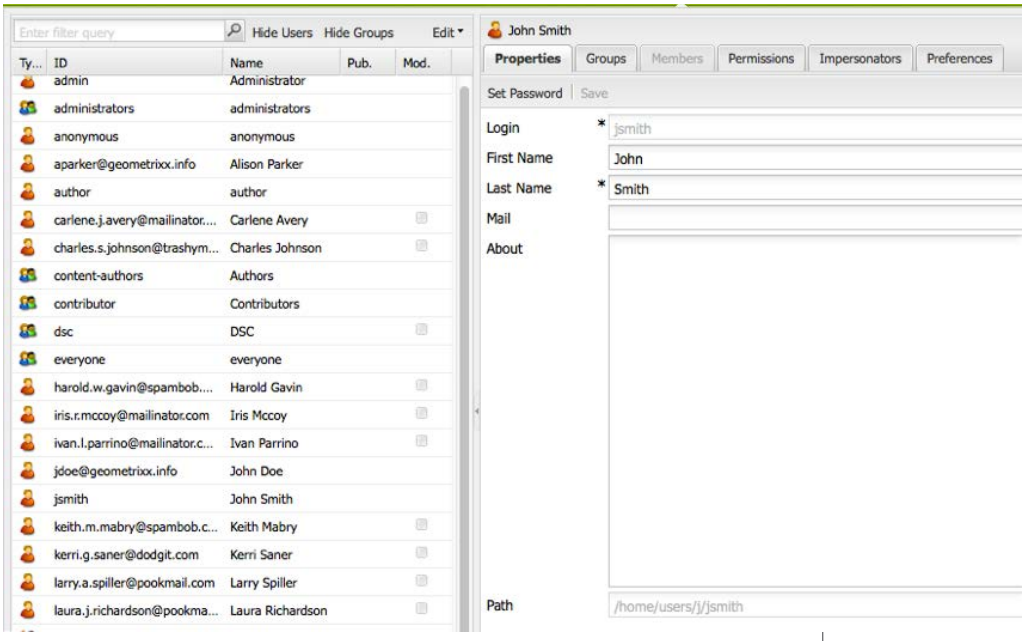
Create a New User

2. The **Create User** dialog box appears. Enter the required details and click **Create**:



Create User Dialog Box

3. Double-click your user icon in the left pane. Notice that it is brought into context in the right pane.



Double-Click the New User for Editing Purposes

4. Click the **Groups** tab. Notice that you do not belong to any group.

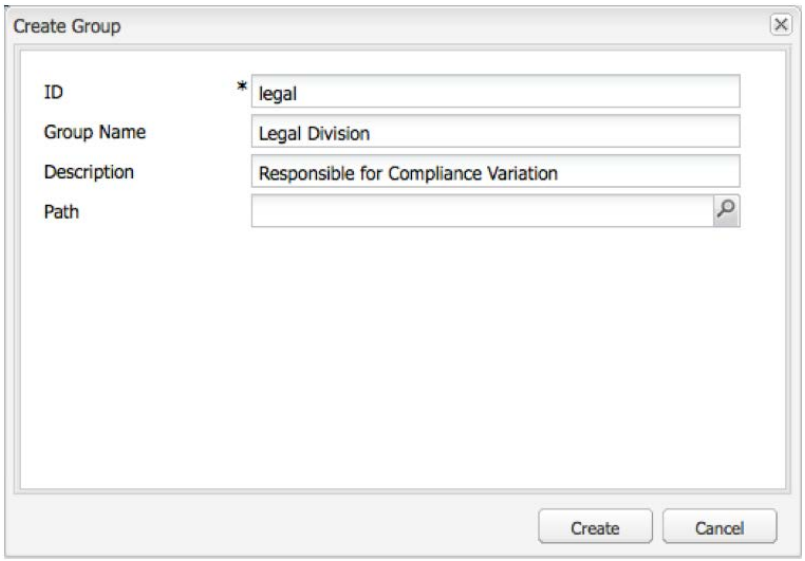
5. Click the **Permissions** tab. You will notice that you have no access to any part of the website. The default permissions policy in AEM is **deny all**.
No users are specified as potential impersonators of you.

Exercise 15.2
Create a Group

Start by creating a new group called Legal.

1. In the **Security** window tree list, click **Edit** > **Create** > **Create Group**.

2. Enter the required details and click **Create**:

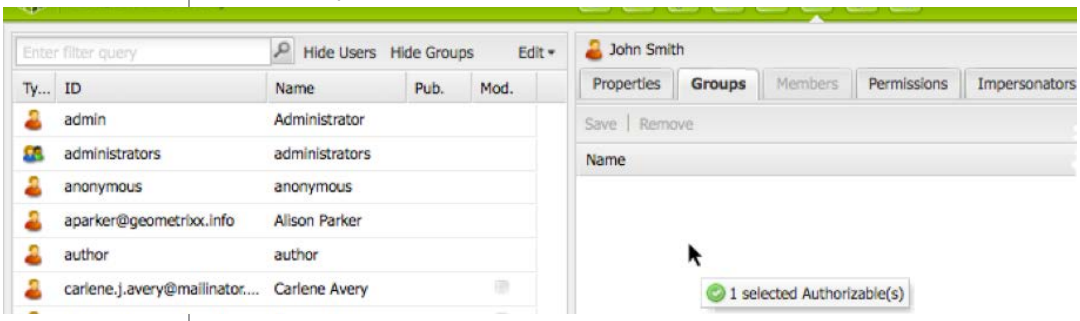


Group Creation Dialog Box

Exercise 15.3
Add a User and Groups to a Group

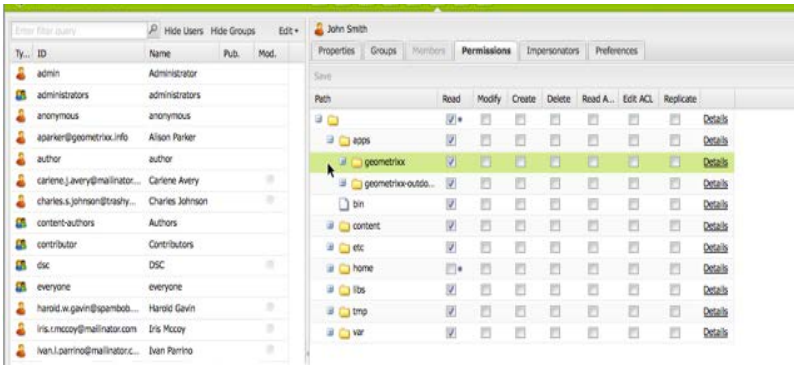
The Groups tab shows you which groups the current account belongs to. You can use it to add the selected account to a group.

1. Double-click the name of your account to assign it to the **Legal** group.
2. Click the **Groups** tab. Now, you will see a list of groups that you already belong to. It is currently empty.
3. In the tree list, click the **Legal** group and drag it to the **Groups** pane. (If you want to add multiple groups, **Shift+click** or **Ctrl+click** those names and drag them.).



Drag the Legal Group to the Groups Pane

4. Click **Save**.
5. Repeat steps 1-4 to add the **Legal** group as a member of the **Contributors** group.
6. Check your **Permissions** tab. Why do you now have access to the Geometrixx website?



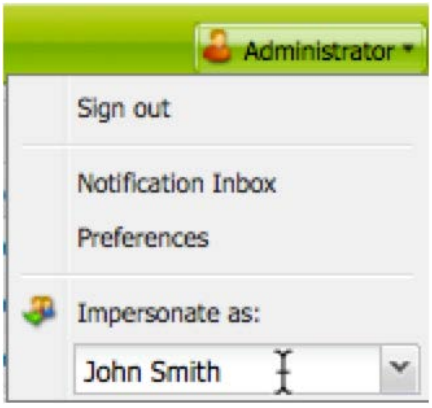
Displaying the Permissions Tab

Now, the user and groups settings are done.

Exercise 15.4
Impersonate a User

You could log out as admin and re-login as another user. It is much easier to impersonate a user to test its settings.

1. Select the **Websites** console.
2. In the top right corner of the WCM view, open the drop-down menu by the word **Administrator**. This menu allows control over logged-in users.
3. In the **Impersonate as** box, choose your user.



Box to Select a User to Act on Your Behalf

The current user is changed to your account.

User on Your Behalf

After you browsed some pages and tested the settings you did, you can finish impersonation by clicking the impersonated user's name and selecting **Revert to self**.

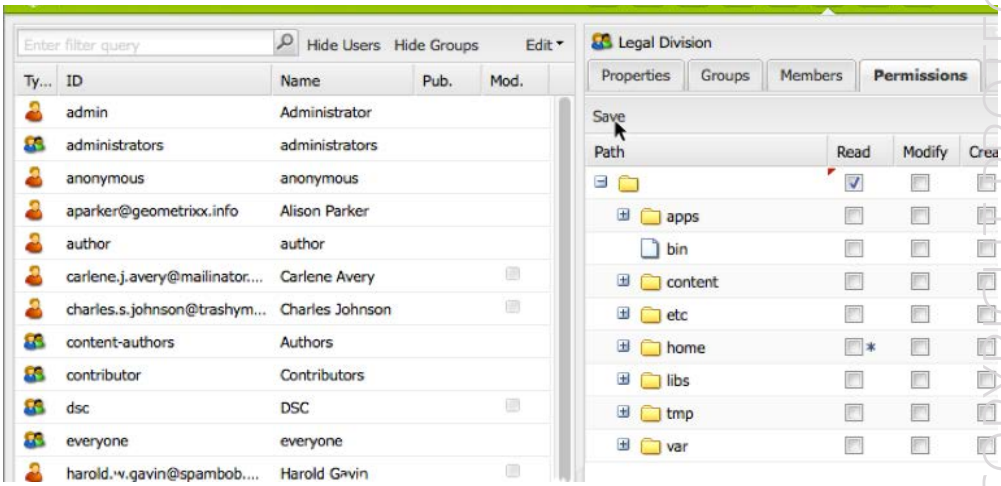
Congratulations! You successfully created users and groups and learned to manage those users and groups. Adobe's best practices suggest that permissions and privileges be granted and denied on a group basis. Groups tend to remain stable, whereas users come and go more frequently.

WARNING: Keep in mind that you will be managing permissions for the newly created Legal group only; although you can manage permissions for any other group or groups, be careful to restore them to their original values because you will need these permissions for the following exercises.

Exercise 15.5
Manage Permissions

To add, modify, or delete page permissions, which enable you to allow or deny the right to perform actions on specific resources:

1. Double-click the **Legal** group to bring it in to context in the right pane.
2. Click the **Permissions** tab. The tree map opens.
3. It is a good idea to provide general read access to the entire repository. Project-specific restrictions can be easily added later. Select the **root** node. By default, users have all access rights denied. To provide read access to the root node, mark the check box in the column Read. Because access rights are automatically inherited to child nodes, all members of the Legal group have now read access to all nodes in the CRX repository.
4. Click **Save**.



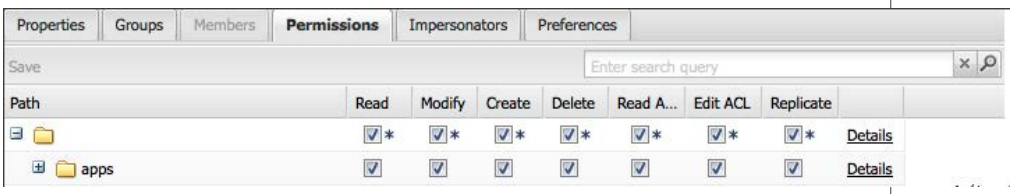
Define the Access Rights

Permissions and ACLs

AEM uses ACLs to determine what actions a user or group can take and where it can perform those actions.

Permissions and ACLs

Permissions define who is allowed to perform which actions on a resource. The permissions are the result of access control evaluations. You can change the permissions granted/denied to a given user by selecting or clearing the check boxes for the individual AEM actions. A checkmark indicates that an action is allowed. No check mark indicates that an action is denied.



The location of the checkmark in the grid also indicates what permissions users have in what locations within AEM (that is, which paths).

Action	Allow (Checkmark)	Deny (No Checkmark)
Description	AEM allows the user to perform the action on this page or on any child pages.	AEM does not allow the user to perform the action on this page or on any child pages.

The permissions are also applied to any child pages. If a permission is not inherited from the parent node but has at least one local entry for it, then the following symbols are appended to the check box. A local entry is one that is created in the CRX 2.3 interface (Wildcard in the path of ACLs, currently can only be created in CRX.)

For an action at a given path:

* (asterisk)	There is at least one local entry (either effective or ineffective). These wildcard ACLs are defined in CRX.
! (exclamation mark)	There is at least one entry that currently has no effect.

When you hover over the asterisk or exclamation mark, a tool tip provides more details about the declared entries. The tool tip is split into two parts:

Upper part	List the effective entries.
Lower part	List the noneffective entries that may have an effect somewhere else in the tree (as indicated by a special attribute present with the corresponding ACE limiting the scope of the entry). Alternatively, this is an entry whose effect has been revoked by another entry defined at the given path or at an ancestor node.

Actions

Actions can be performed on a page (resource). For each page in the hierarchy, you can specify which action the user is allowed to take on that page. Permissions enable you to allow or deny an action.

Action	Description
Read	The user is allowed to read the page and any child pages.
Modify	The user can: • Modify existing content on the page and on any child pages. • Create new paragraphs on the page or on any child page. At the JCR level, users can modify a resource by modifying its properties, locking, visioning, and nt-modifications, and they have complete write permission on nodes defining a jcr:content child node, for example, cq:Page, nt:file, and cq:Asset.
Create	The user can create a new page or child page. If modify is denied, the subtrees below jcr:content are specifically excluded as the creation of jcr:content and its child nodes are considered a page modification. This only applies to nodes defining a jcr:content child node.
Delete	The user can: • Delete existing paragraphs from the page or any child page. • Delete a page or child page. If modify is denied, any sub-trees below jcr:content are specifically excluded as removing jcr:content and its child nodes is considered a page modification. This only applies to nodes defining a jcr:content child node.
Read ACL	The user can read the ACL of the page or child pages.
Edit ACL	The user can modify the ACL of the page or any child pages.
Replicate	The user can replicate content to another environment (for example, the Publish instance). The privilege is also applied to any child pages.

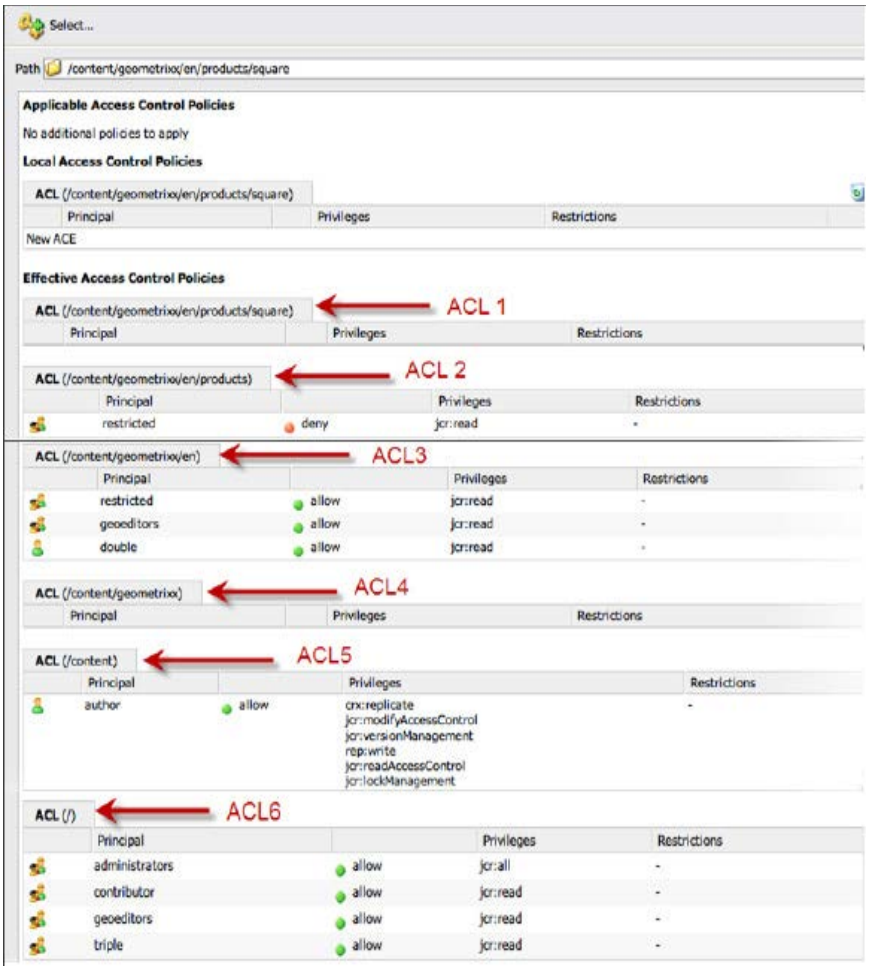
ACLs and How They Are Evaluated

AEM WCM uses ACLs to organize the permissions being applied to the various pages. ACLs are made up of individual permissions and are used to determine the order in which these permissions are actually applied. The list is formed according to the hierarchy of the pages under consideration. This list is then scanned bottom-up until the first appropriate permission to apply to a page is found.

Assume that a user wants to access the following page:

/content/geometrixx/en/products/square

and the ACL list for that page is the following:



The ACL (or permission) that will be applied to the page is ACL 1, related to /content/geometrixx/en/products/square. In this case, the ACL associated to this page is empty, so AEM will go up one level to ACL 2.

ACL 2 will be applied if the user belongs to the restricted group, and in that case, the user will have access denied for this page. If the user is not part of the restricted group, then AEM will go up another level to ACL 3.

ACL 3 will be applied if the user belongs to the group geoeditors or double only; in this case, the user will have the access granted to the page. If the user is part of the restricted group, as you saw, the ACL applied is ACL 2. If the user is not part of the three mentioned groups before, AEM will go up one level to ACL 4.

With ACL 4 being empty, if AEM reaches this level, it will go up one level again to ACL 5.

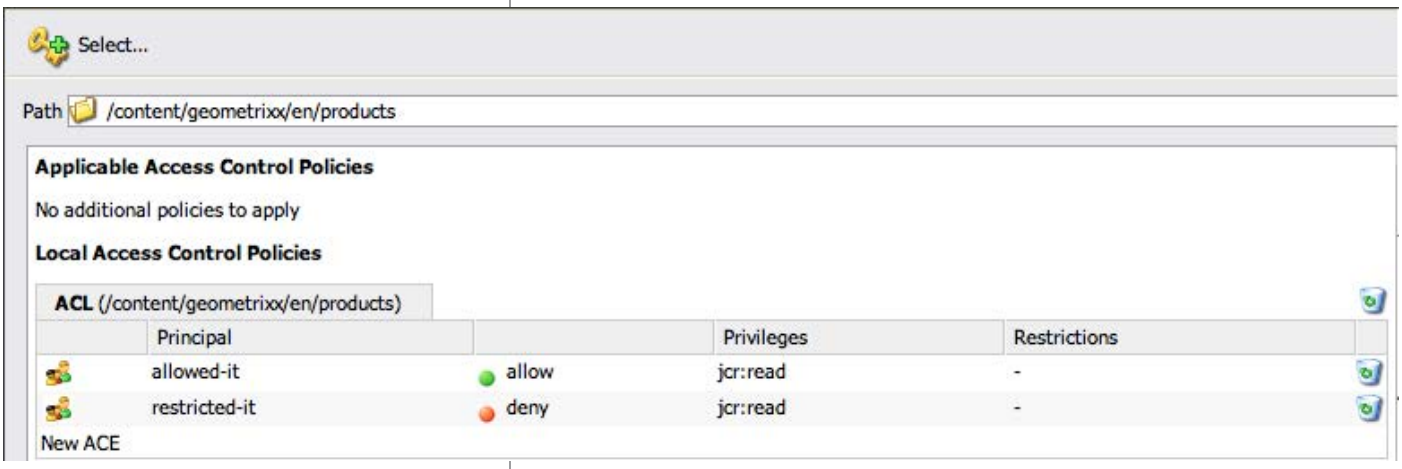
ACL 5 will be applied if the user belongs to the author group, and then the user will have the access granted to the page. If the user is not part of the author group, AEM will go up one level to ACL 6.

ACL 6 will be applied if the user belongs to the administrators, contributor, or triple group; if this is the case, the user will have access granted to the page. If the user is part

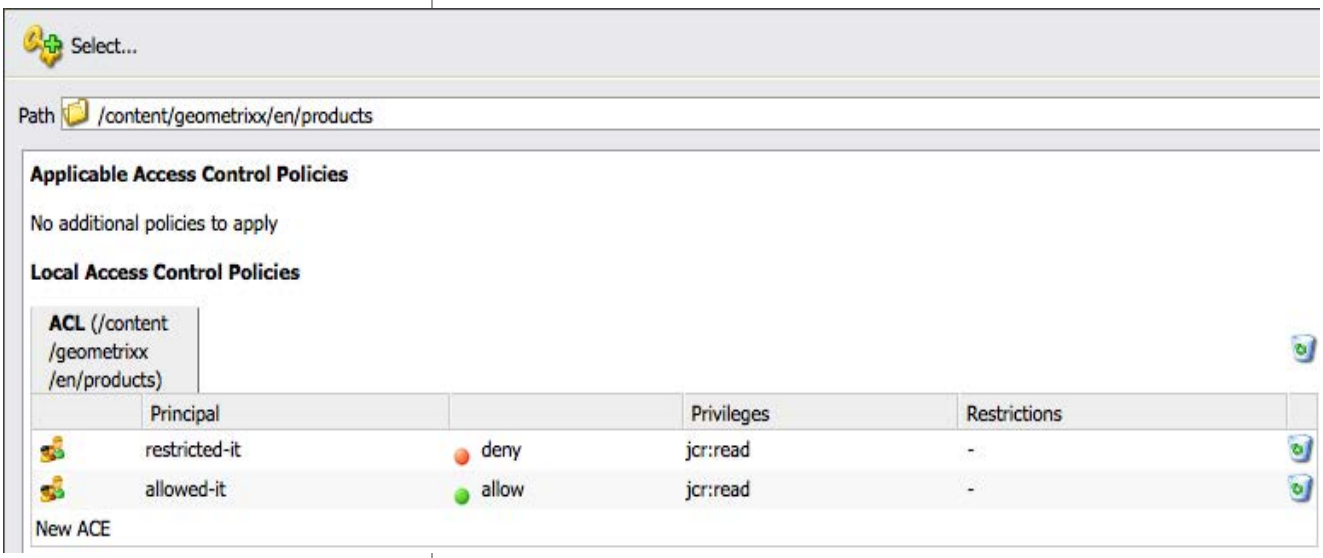
of the geoeditors group as you saw, then AEM would have had already applied ACL 3 and granted access to the page. If the user is not part of any group, the access to the page is denied.

Concurrent Permission on ACLs

When two concurrent (and opposing) permissions are listed on the same ACL for the same resource, the ACL that is applied for such a resource is the one at the bottom. Suppose, you have the following ACL for the same resource under /content/geometrixx/en/products.



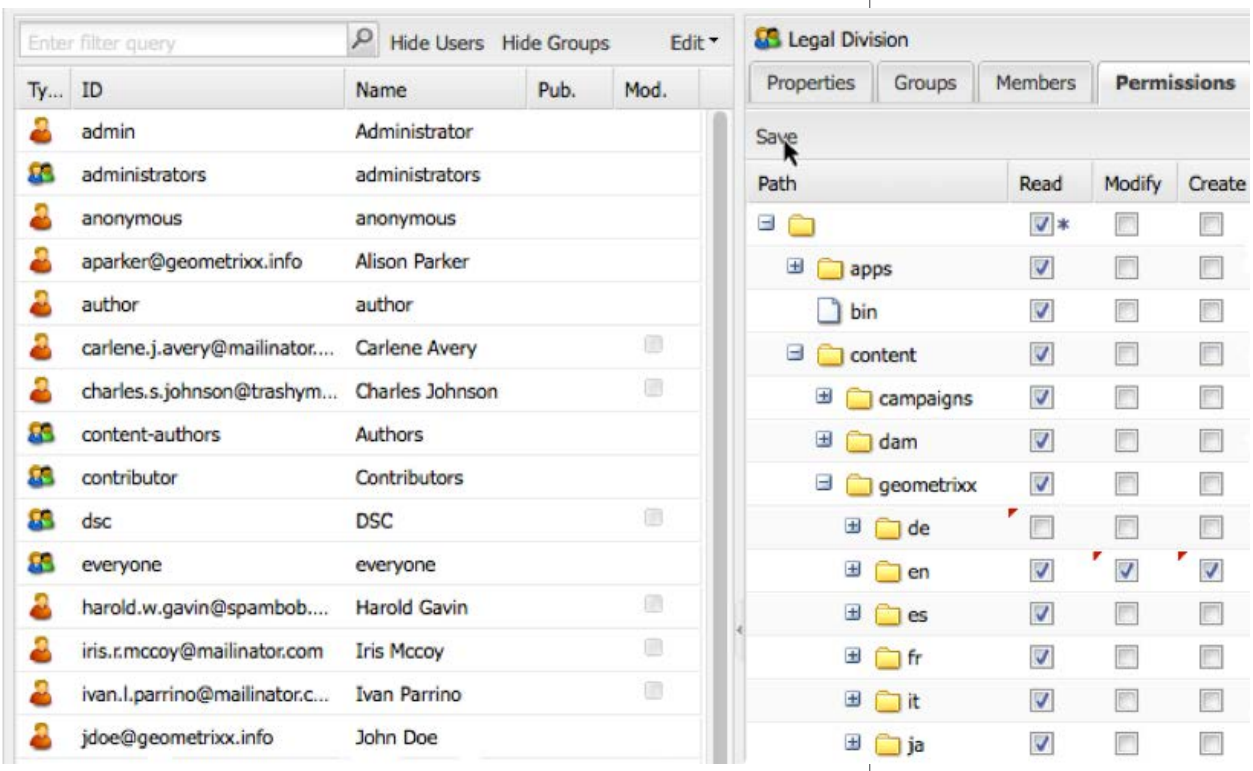
If a user is part of the two groups 'allowed-it' and 'restricted-it', you can see that the access to the page products is denied because the ACL deny in read access is the rule at the bottom.



Now, if the order of the ACL is the opposite, a user, part of two groups, allowed-it and restricted-it, will have access to the products page (because the ACL allow in read access is the rule at the bottom).

Exercise 15.6
Manage Access Rights for Different Websites

1. Check the check box in the **Modify** column for the start page of the branch you want to provide access to. In our case, **/content/geometrixx/en**.
2. Do the same for the **Create** column. The red corner indicates that the item listed has not yet been saved.
3. Click **Save**.
4. Navigate to **/content/geometrixx/de** and clear the check box in the **Read** column.
5. Click **Save**.



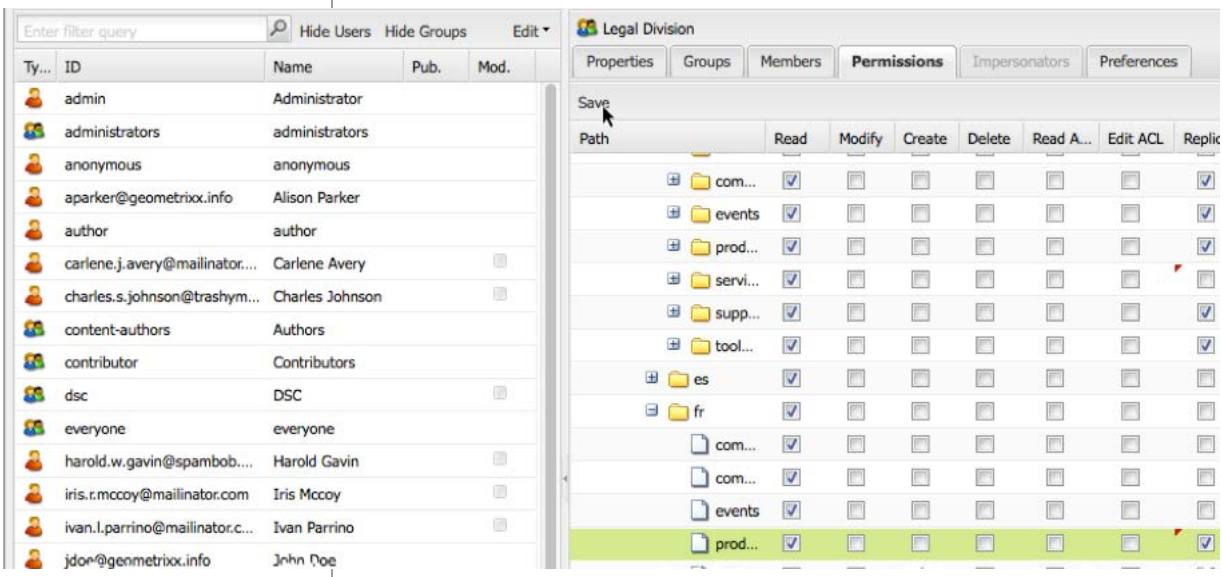
Manage Access Rights for the Website

Exercise 15.7
Manage Replication Privileges

Replication privileges are the right to activate or deactivate content, and can be set for groups and users.

1. Select the **Replicate** check box of the node **/content/geometrixx/en**. Save to allow all pages below to be replicable too.
2. Clear the **Replicate** check box of **/content/geometrixx/en/services**. Save.
3. Now, modify the replication privileges for the **French** branch.
4. Select the **Replicate** check box of the node **/content/geometrixx/fr/products**.

5. Click **Save**.



Defining Access Rights to Replicate Pages

As you can see, you can provide fine-grained replication privileges not only for an entire tree branch, but also at page level.

Users without replication privilege granted still have access to the **Activate/Deactivate** buttons. Clicking them will not have the desired effect immediately. Instead, a workflow is started, which puts the requested action in the inbox of privileged users requesting them to approve and finish the action.

Exercise 15.8
Deny Access Rights to Consoles

You want to deny access to different AEM consoles by simply hiding the icons in the main view. The table below shows every icon with the corresponding node; you need to deny read access. You can deny access to any of the console buttons, but do not forget to display at least one. Otherwise, users cannot log in. Usually, it is the SiteAdmin console, which is accessible to everybody. The restrictions for different websites have already been done in the previous section.

Console	Node Path in CRX
Site Admin (Websites)	/libs/wcm/core/content/siteadmin
DAM Admin	/libs/wcm/core/content/damadmin
Tools	/libs/wcm/core/content/misc

NOTE: Be careful when denying access to consoles. If you deny access to all consoles, the user will not be able to successfully complete the login process.

Console	Node Path in CRX
Security (Users)	/libs/cq/security/content/admin
Inbox	/libs/cq/workflow/content/inbox
Tagging	/libs/cq/tagging/content/tagadmin
Campaigns	/libs/mcm/content/admin
Community	/libs/collab/core/content/admin

Remove Access to the Navigation Option in the Rail

1. Open the user and/or group management and select the user/group you want to restrict access for.
   Avoid assigning/restricting permissions on a user-by-user basis. It is recommended to use groups.
2. Remove access permissions to the appropriate node(s) under **/libs/cq/core/content/nav/sites**. These correlate to the navigation options in the Rail:
   - Projects
   - Sites
   - Apps
   - Publications
   - Forms
   - Assets
   - Communities
   - Commerce
   - Tools

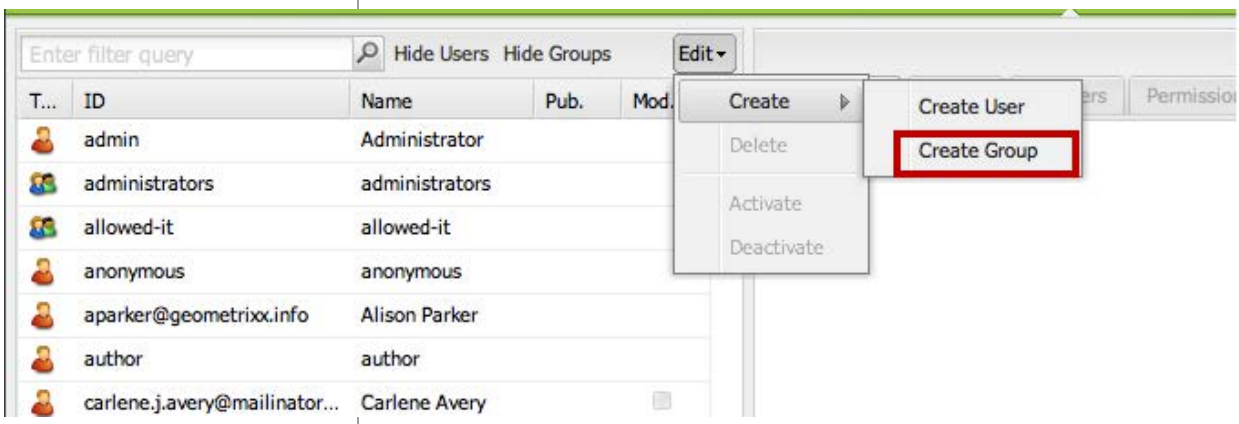
Exercise 15.9
Manage Conflicting ACLs

Goal

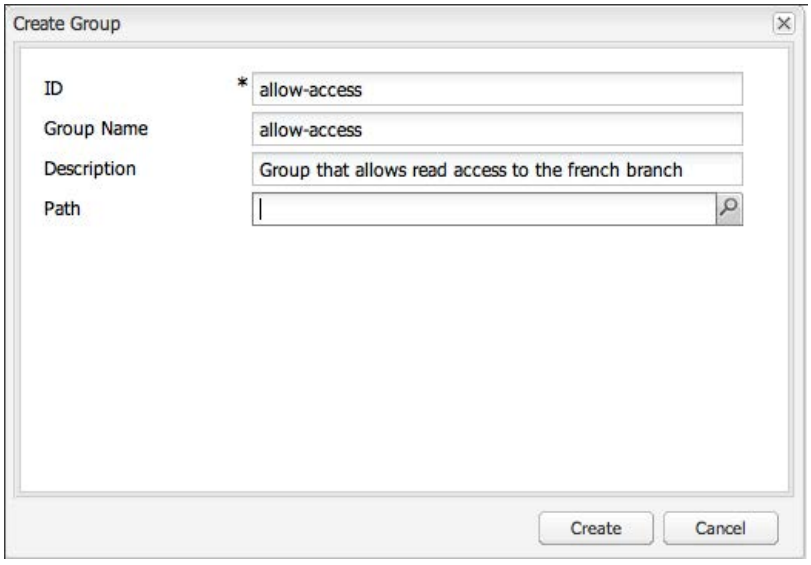
You are going to create a user who will be part of two groups with two different ACLs, and you will modify them programmatically.

Steps

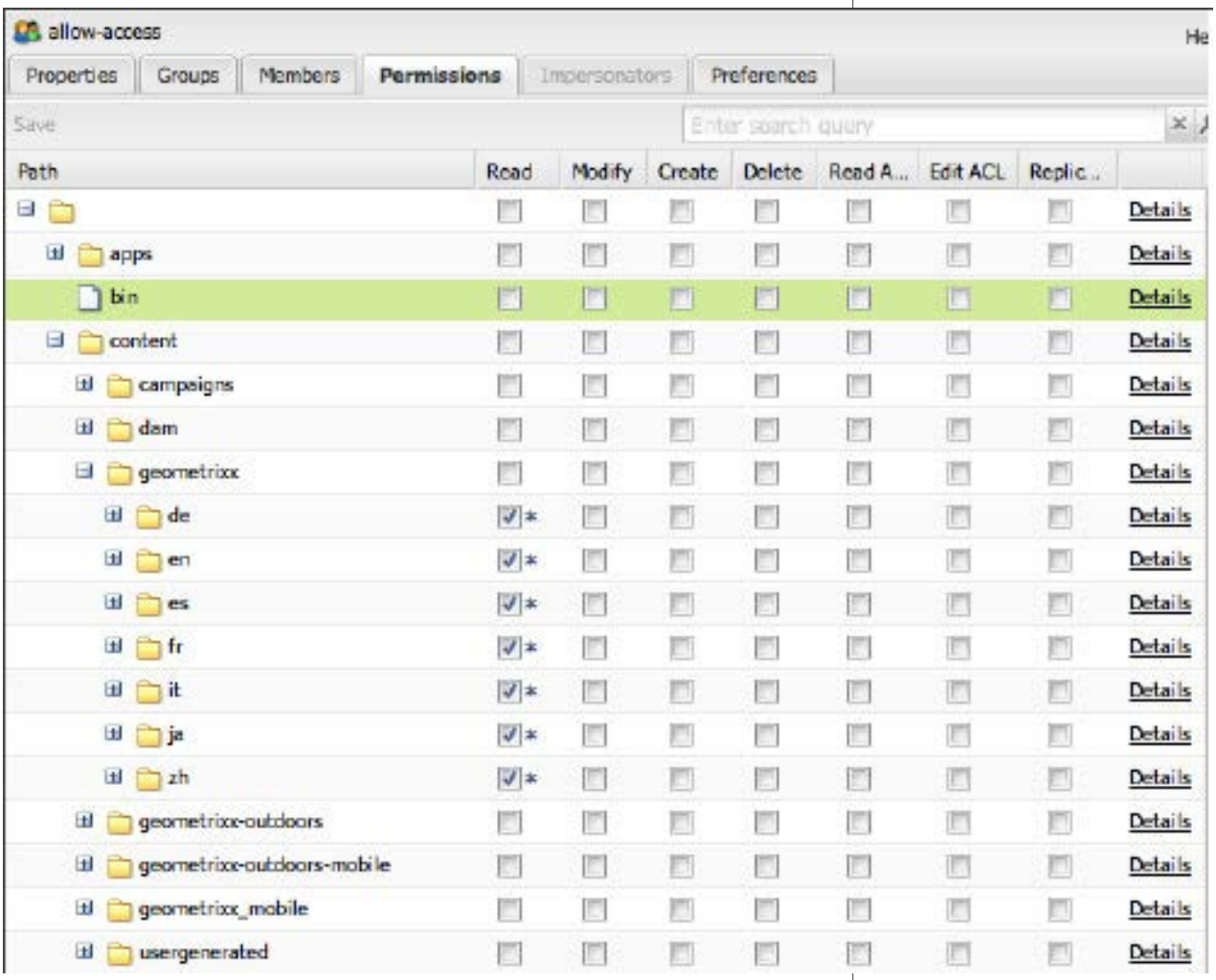
1. Create a group named **allow-access**; the group should have read access to **/content/geometrixx/fr**. Choose the user menu, then on the left pane, click **Edit**. A drop-down menu will allow you to create your group.



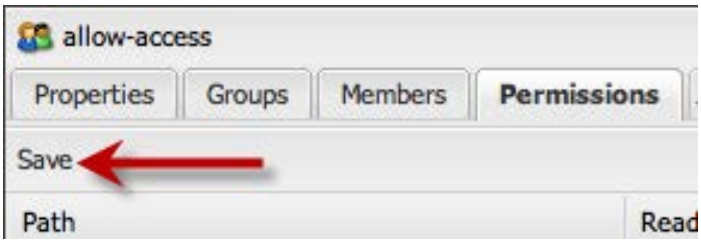
Create the group like the following:



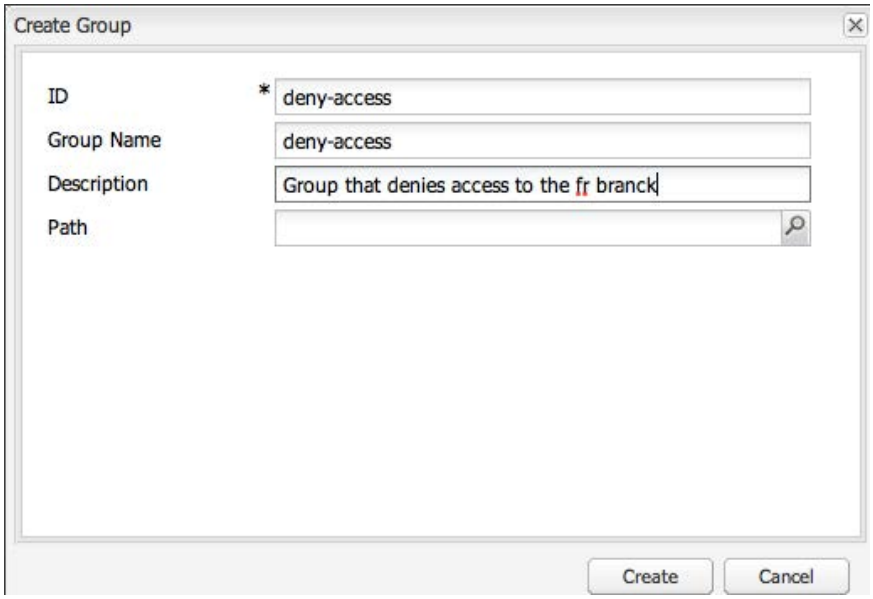
Open the newly created group and assign read rights to all language branches like the following:



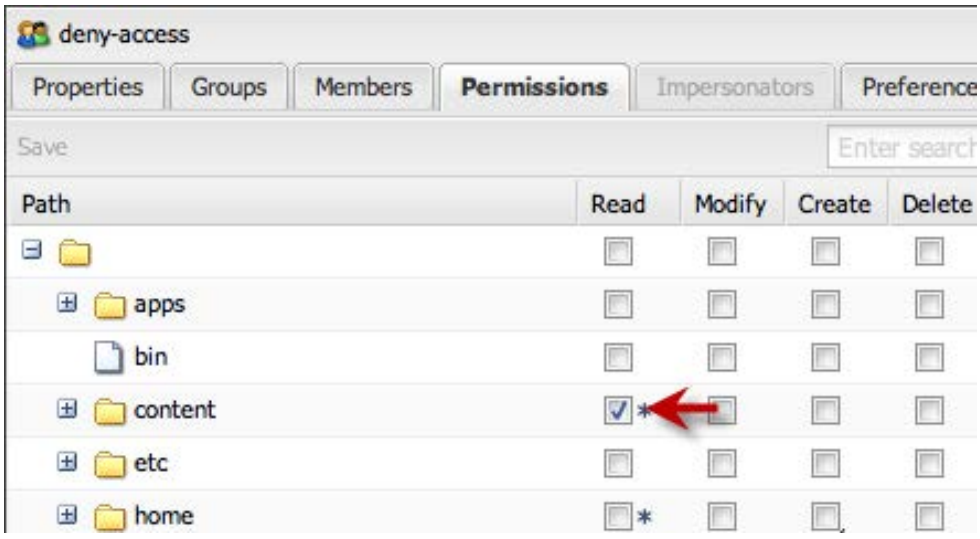
Don't forget to click on **Save** after you are done.



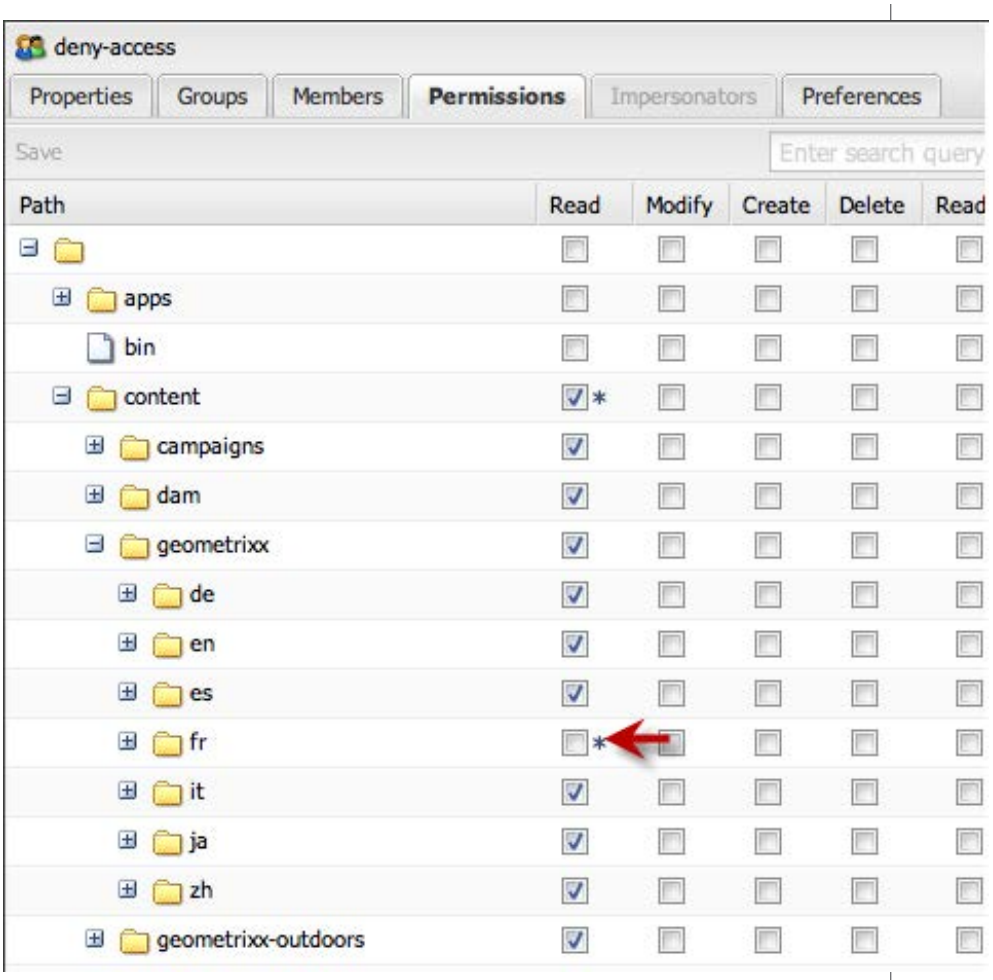
2. Create another group, **deny-access** The group should not have read access to **/content/geometrixx/fr**.



Open the newly created group and assign read rights to the all the language branches, except for the **French** one, like the following.
To do that, first provide read access to the **/content** node.

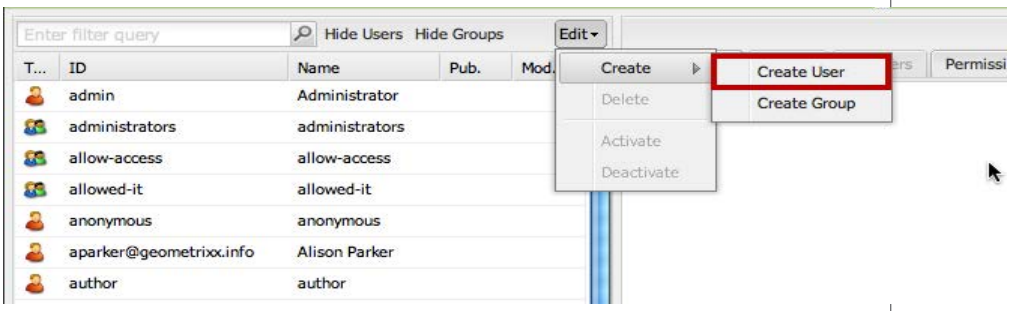


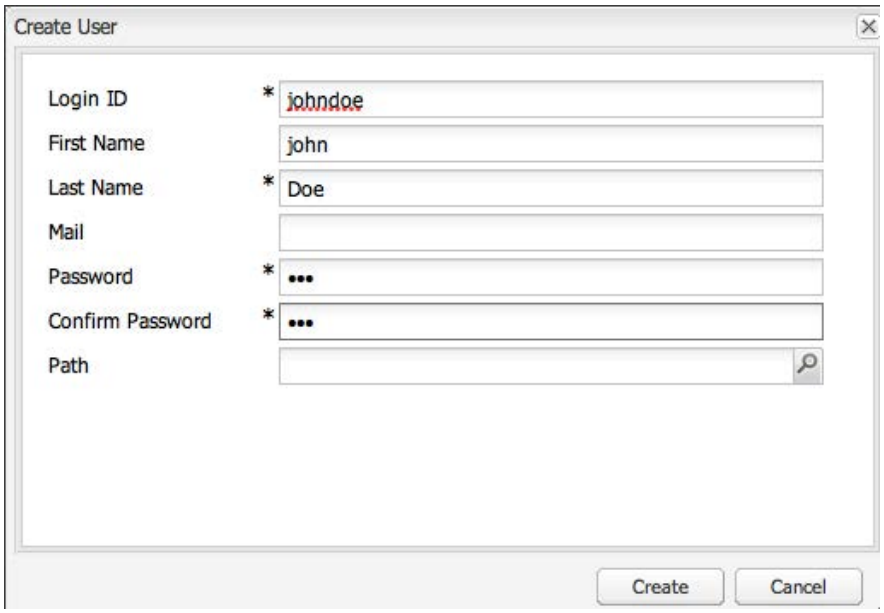
Save your changes, refresh your page, expand the content node, and clear the **fr** node.



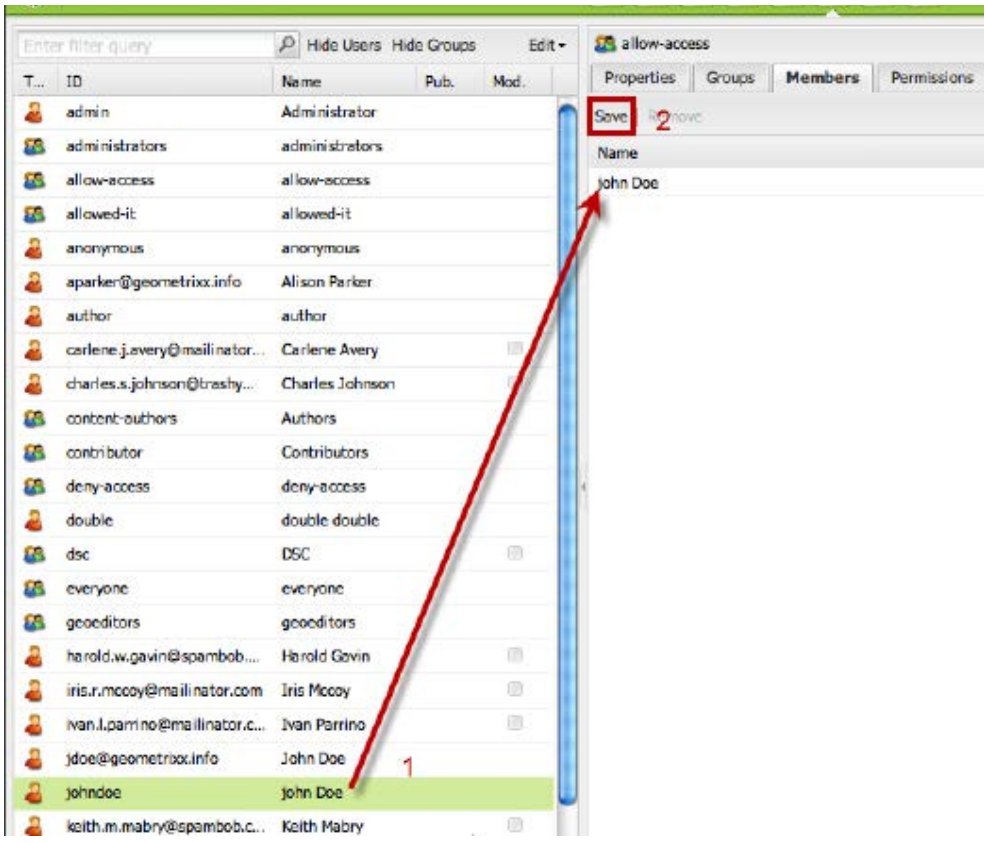
Do not forget to save the changes by clicking **Save.**

3. Create a user, **John Doe**; to do that, proceed the same way as you did when creating a new group, but choose **Create User**.





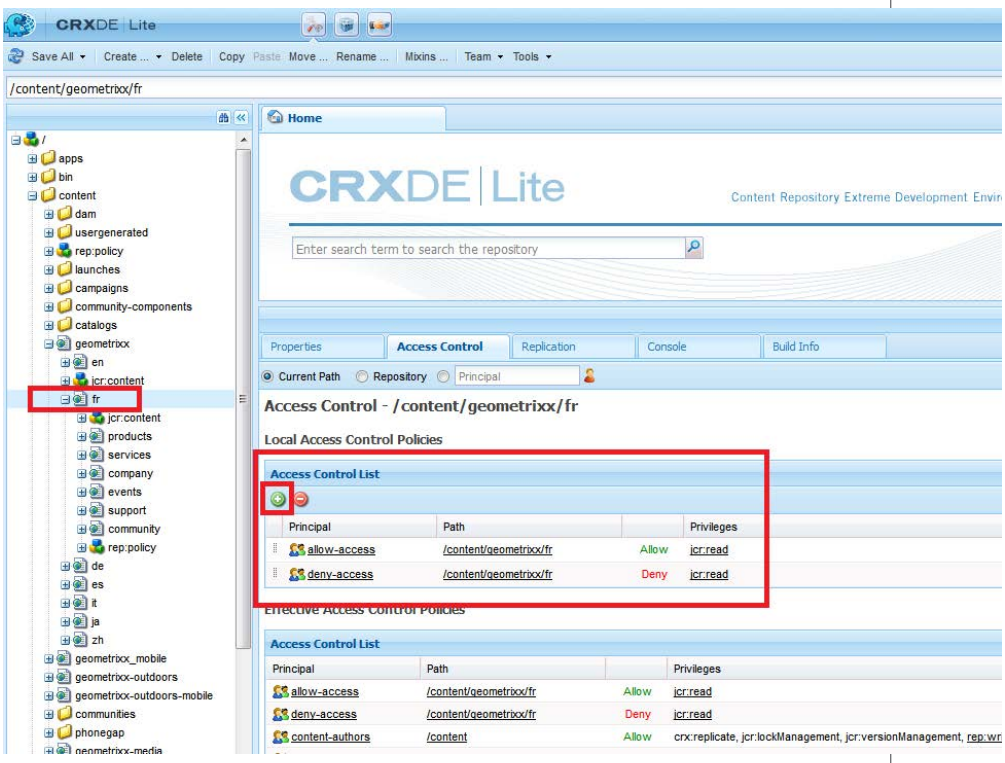
4. Add the user **John Doe** to the **allow-access** and **deny-access** group. For that, you will have to open each group, click **Members**, and drag the user **John Doe** to the member list.



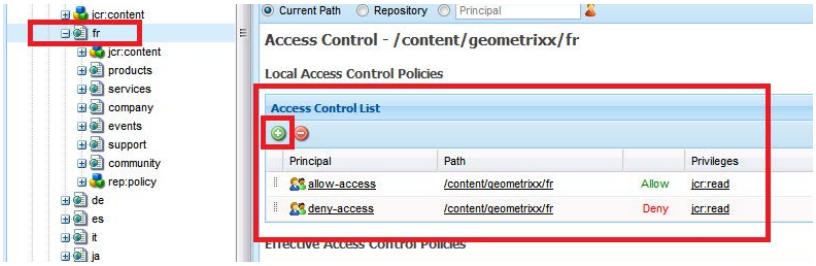
Do not forget to click **Save** each time.

5. Add the user to the group contributors so that the user also has access to elements like CSS-JS libraries and the design associated to the page.

6. Open CRXDE Lite and click the **French** node, then in the **Access control List**, click the + sign to add both lists.



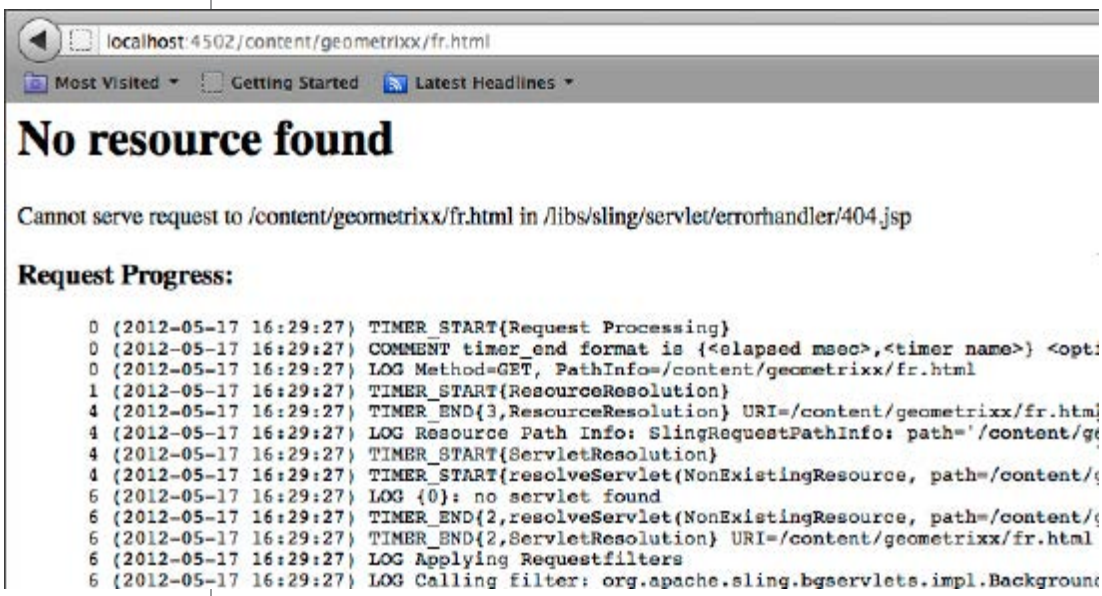
You should see the following information:



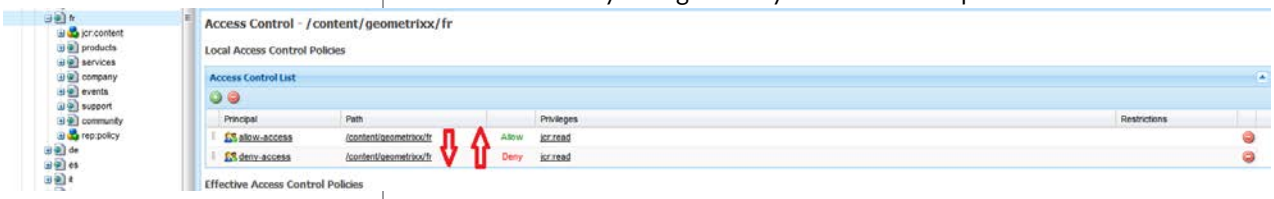
As you can see, the user John Doe does not have access to the French page because the deny rule is the one at the bottom of the ACL rules applied to the /content/geometrixx/fr resource and therefore takes precedence.

7. Open http://localhost:4502/content/geometrixx/fr.html in your browser.Log in as John Doe.

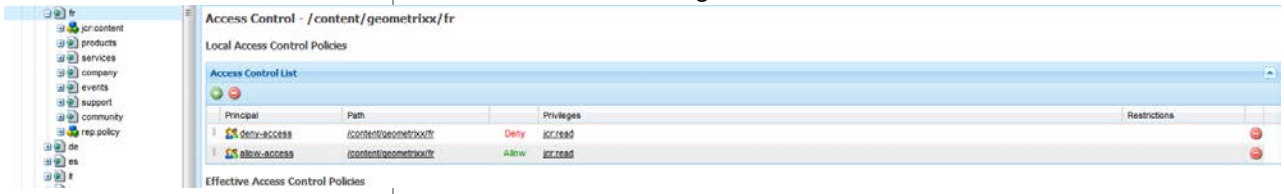
As John Doe does not have access to the page, you should see the following:



8. Open the ACL in CRXDE Lite for the **French** page. You will change the order of the ACL list by sliding the deny access ACL on top of the allow access ACL.



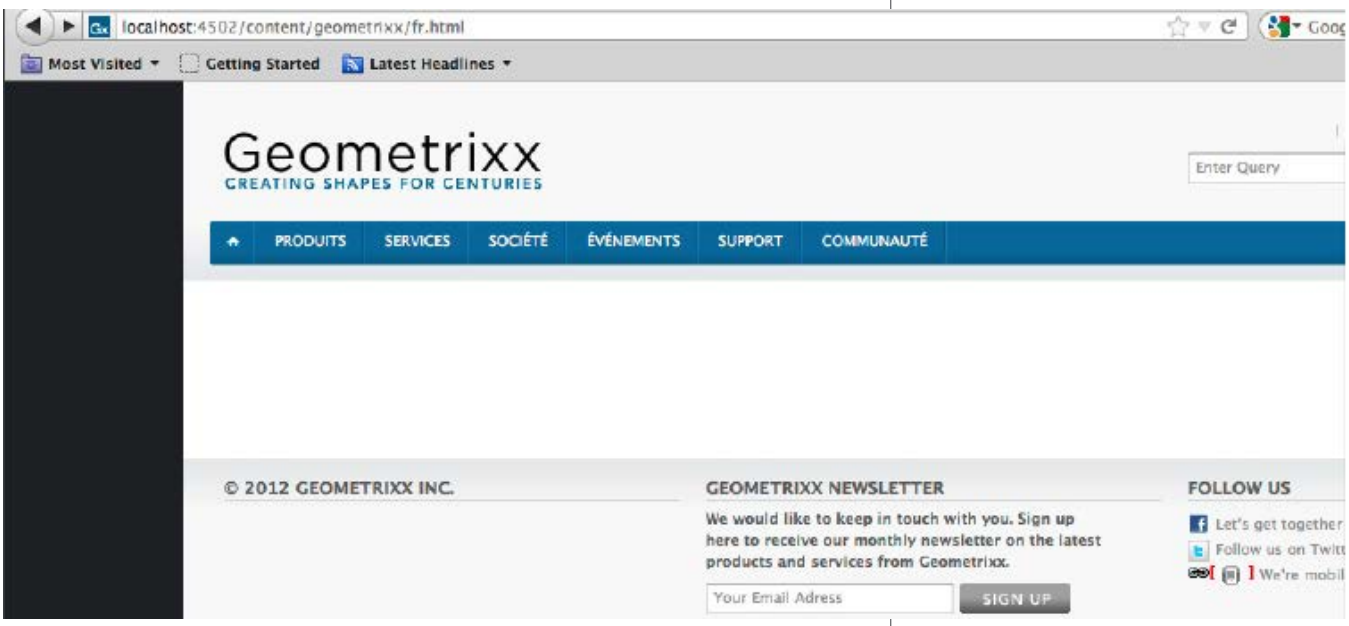
You should see the following:



9. Click the **Apply** button, and then click **OK**.

10. John Doe now should have access to the French page as the ACL rule on the bottom is the one granting him read access to the node. Disconnect from the previous session and open http://localhost:4502/content/geometrixx/fr.html. Again, use **John Doe** to log in.

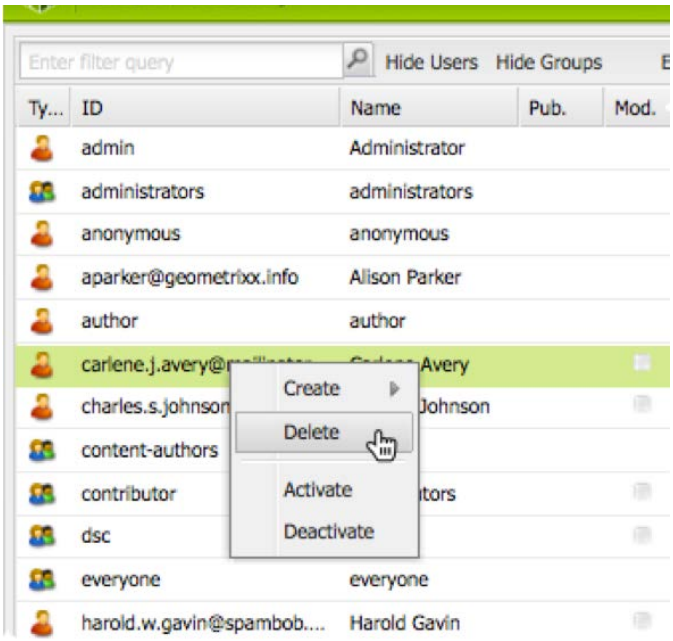
This time you should be able to see the French page.



Exercise 15.10
Delete Users or Groups

To delete a user or a group:

1. In the **Security** window, select any user or group. If you want to delete multiple items, **Shift+click** or **Ctrl+click** to select them.
2. Click **Edit** or right-click the item to bring up the context menu. Select **Delete**. You will be prompted to confirm the delete action.
3. Click **OK** to confirm.



Delete a user from the AEM

The online documentation http://dev.day.com/content/docs/en/cq/current/administering/security.html contains more information related to user and group management.

Integrating With LDAP and Enabling Single Sign On

Goal

LDAP is used for accessing centralized directory services. This helps reduce the effort required to manage user accounts as they can be accessed by multiple applications. One such LDAP server is Active Directory. LDAP is often used to achieve Single Sign On, which allows a user to access multiple applications after logging in once.

LDAP authentication is required to authenticate users stored in a (central) LDAP directory such as Active Directory. This helps reduce the effort required to manage user accounts. User accounts can be synchronized between the LDAP server and CRX, with LDAP account details being saved in the CRX repository. This allows the accounts to be assigned to CRX groups for allocating the required permissions and privileges.

CRX uses LDAP authentication to authenticate such users, with credentials being passed to the LDAP server for validation, which is required before allowing access to CRX. To improve performance, successfully validated credentials can be cached by CRX, with an expiry timeout to ensure that revalidation does occur after an appropriate period.

When an account is removed from the LDAP server, validation is no longer granted and so access to CRX is denied. Details of LDAP accounts that are saved in CRX can also be purged from CRX.

Use of such accounts is transparent to your users; they see no difference between user and group accounts created from LDAP and those created solely in CRX.

In AEM 6.1, LDAP support comes with a new implementation that requires a different type of configuration than with previous versions.

All LDAP configurations are now available as OSGi configurations. They can be configured through the Web Management console at:

http://localhost:4502/system/console/configMgr

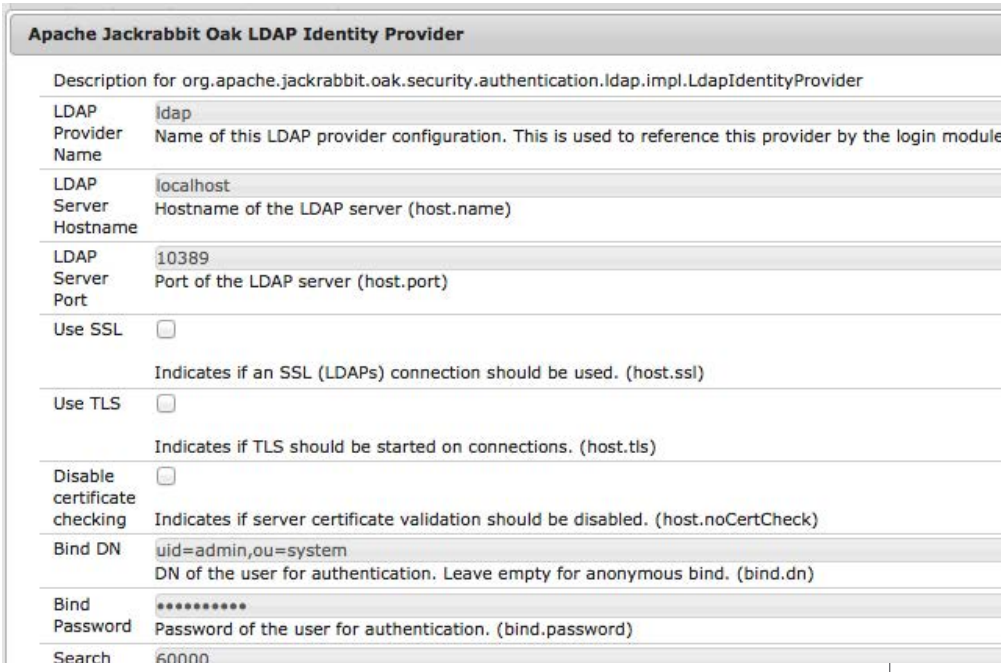
In order to have LDAP working with AEM, you need to create three OSGi configurations:

1. An LDAP Identity Provider (IDP)
2. A synchronization handler
3. An external login module

Configuring the LDAP Identity

The LDAP IDP is used to define how users are retrieved from the LDAP server.
It can be found in the management console under the **Apache Jackrabbit Oak LDAP Identity Provider** name.
The following configuration options are available for the LDAP IDP:

LDAP Provider Name	The name of this LDAP provider configuration
LDAP Server Host name	The host name of the LDAP server
LDAP Server Port	The port of the LDAP server
Use SSL	Indicates if an SSL (LDAPs) connection should be used
Use TLS	Indicates if TLS should be started on connections
Disable certificate hacking	Indicates if server certificate validation should be disabled
Bind DN	The DN of the user for authentication; if this is left empty, an anonymous bind will be performed
Bind Password	The password of the user for authentication
Search timeout	Time until a search times out
User base DN	The DN for user searches
User object classes	The list of object classes a user entry must contain
User id attribute	The name of the attribute that contains the user ID
User extra filter	Extra LDAP filter to use when searching for users; the final filter is formatted like: '(&(<idAttr>=<userId>)(objectclass=<objectclass>)<extraFilter>)' (user.extraFilter)
User DN paths	Controls if the DN should be used for calculating a portion of the intermediate path
Group base DN	The base DN for group searches
Group object classes	The list of object classes a group entry must contain
Group name attribute	The name of the attribute that contains the group name
Group extra filter	The extra LDAP filter to use when searching for groups; the final filter is formatted like: '(&(<nameAttr>=<groupName>)(objectclass=<objectclass>)<extraFilter>)'
Group DN paths	Controls if the DN should be used for calculating a portion of the intermediate path
Group member attribute	The group attribute that contains the member(s) of a group



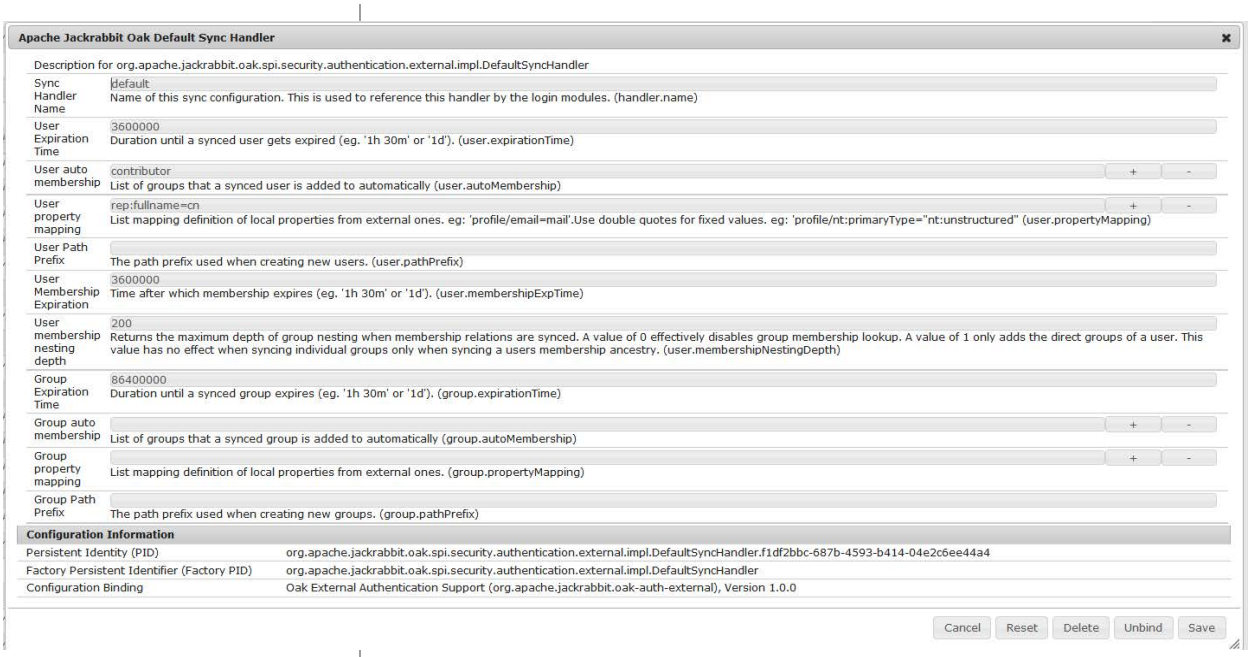
Provider Configuring the Synchronization Handler

The synchronization handler will define how IDP users and groups will be synchronized with the repository.

It is located under the **Apache Jackrabbit Oak Default Sync Handler** name in the management console.

The following configurations options are available for the synchronization handler:

Sync Handler Name	The name of the sync configuration
User Expiration Time	Duration until a synced user is expired
User auto membership	The list of groups that a synced user is added to automatically
User property mapping	The list mapping definition of local properties from external ones
User Path Prefix	The path prefix used when creating new users
User Membership Expiration	Time after which membership expires
User membership nesting depth	Returns the maximum depth of group nesting when membership relations are synced; a value of 0 effectively disables group membership lookup; a value of 1 only adds the direct groups of a user; this value has no effect when synchronizing individual groups only when synchronizing a user's membership ancestry
Group Expiration Time	Duration until a synced group expires
Group auto membership	The list of groups that a synced group is added to automatically
Group property mapping	The list mapping definition of local properties from external ones
Group path prefix	The path prefix used when creating new groups.



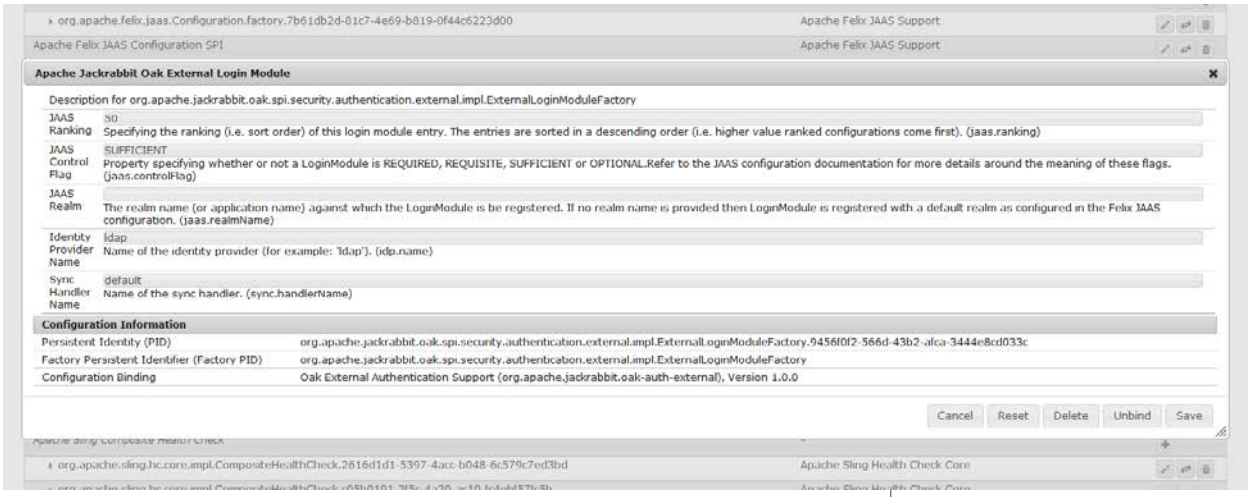
The External Login Module

The external login module is located under the **Apache Jackrabbit Oak External Login Module** under the management console.

Its job is to define which IDP and synchronization handler to use, effectively binding the two modules.

The following configuration options are available:

JAAS Ranking	Specifys the ranking (sort order) of this login module entry; the entries are sorted in a descending order (higher value-ranked configurations come first)
JAAS Control Flag	Property specifying whether a LoginModule is REQUIRED, REQUISITE, SUFFICIENT or OPTIONAL; refer to the JAAS configuration documentation for more details around the meaning of these flags
JAAS Realm	The realm name (or application name) against which the LoginModule is be registered; if no realm name is provided then LoginModule is registered with a default realm as configured in the Felix JAAS configuration
Identity Provider Name	The name of the IDP
Sync Handler Name	The name of the synchronization handler



Exercise 16.1
Install a Local LDAP Server

Installing and configuring an LDAP server is the first step in integrating AEM and an LDAP server. You will use a local LDAP server in this class for convenience.

LDAP Installation Instructions for Windows

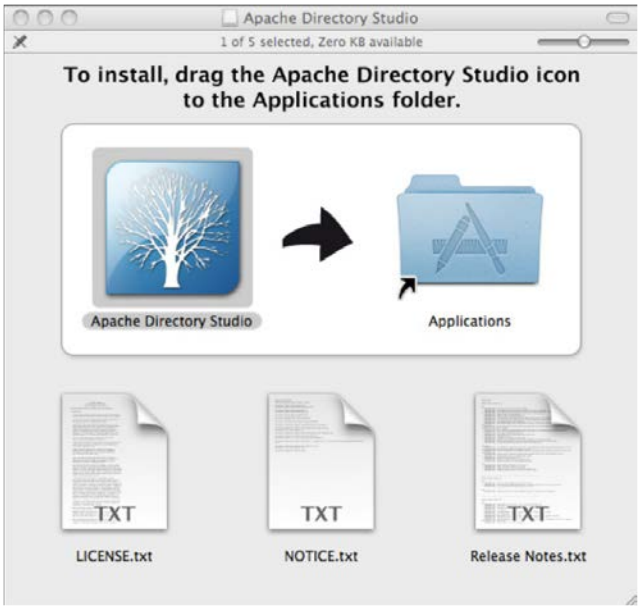
1. In the directory **distribution/ldap** of the training memory stick, you will find a self-extracting executable named **ApacheDirectoryStudio-win32-x86_<architecture>-<version>**. It contains an OpenLDAP server that you will configure to be used with CRX. Choose the executable that matches your Java architecture: 32-bit or 64-bit.
2. Double-click the installer to open it.

3. You may be prompted to confirm your action. If such a popup is displayed, to open the installer, click **Run**.
4. Click **Next** to follow the instructions and complete the installation. See the next section for instructions on defining/configuring the LDAP server.

NOTE: If you plan to have more than one LDAP configuration with your AEM instance, separate IDPs and synchronization handlers need to be created for each configuration.

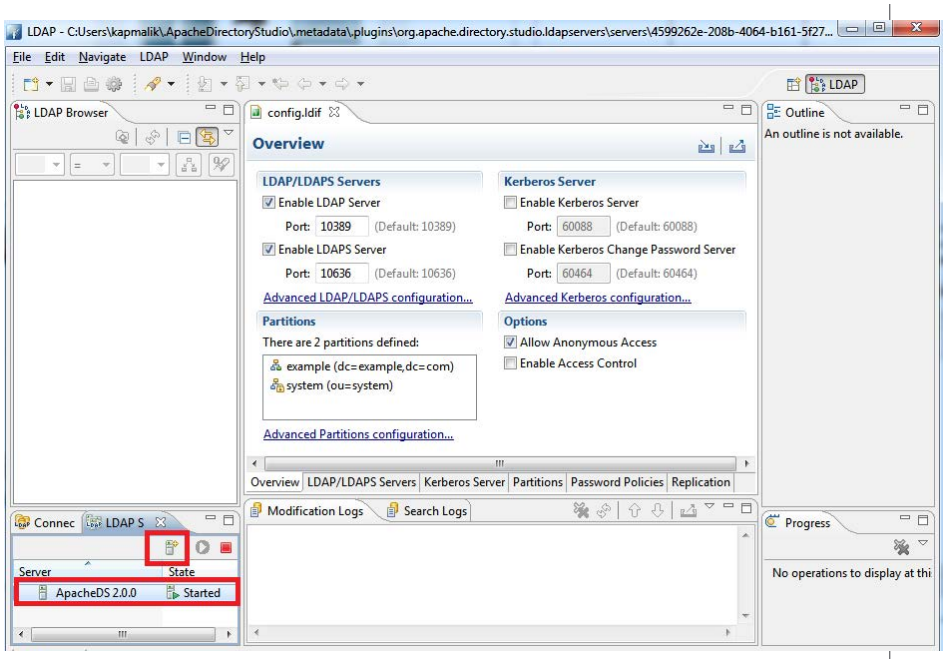
LDAP Installation Instructions for Mac OS X

1. In the directory **distribution/ldap** of the training memory stick, you will find a DMG archive named **ApacheDirectoryStudio-macosx-<version>.dmg**. It contains an Open LDAP server that you will configure to be used with CRX.
2. Double-click the DMG file to start the installation process. See the next section for instructions on defining/configuring the LDAP server.



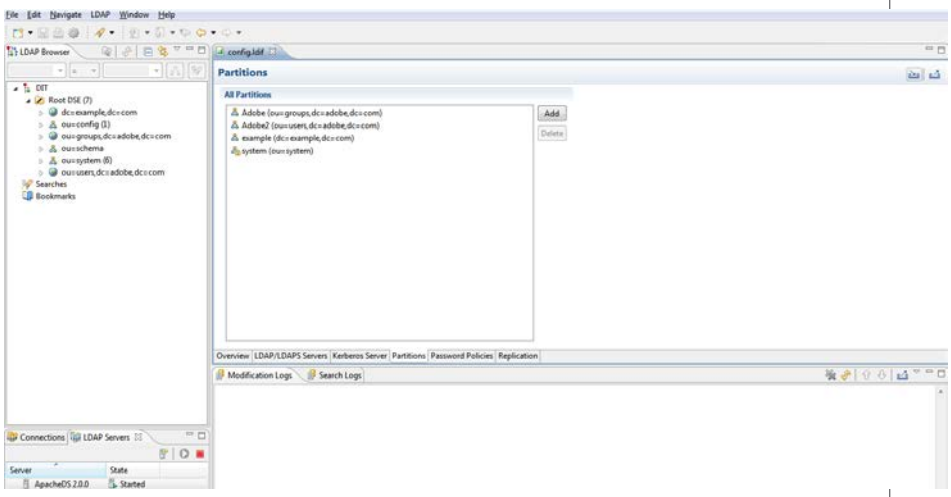
Exercise 16.2
Set Up/Configure the LDAP Server

1. When installed, open Apache Directory Studio. Go to the Server tab and create a new server with the name AEM6LDAP where the type of server will be ApacheDS 2.0.0. Go to the LDAP tab displayed at the bottom-left corner. Click the New Server icon to create a new server.

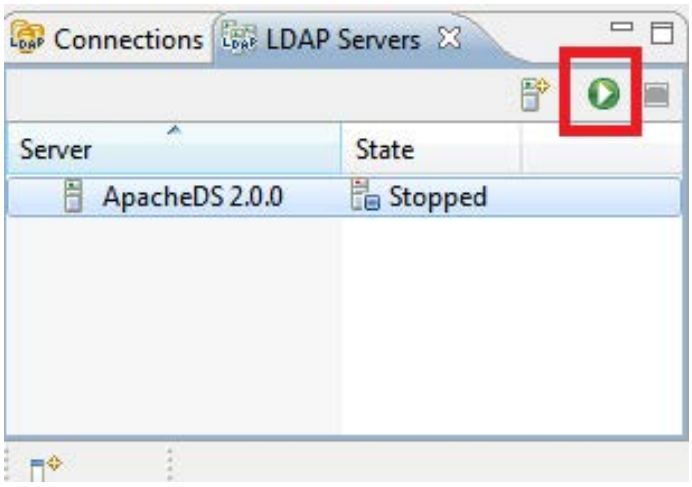


2. When the server is created and started, double-click the AEM6LDAP server and you will see the config.ldif overview tab opened with the default configuration as shown in the screenshot above.

3. Access the **partitions** tab of the server configuration and click **Advanced Partition Configuration**. Click **Add** and enter partitions for groups (name: **Adobe** and suffix: **ou=groups,dc=adobe,dc=com**) and for users (name: **Adobe2** and suffix: **ou=users,dc=adobe,dc=com**). Save by choosing **File > Save**. Restart the server by stopping and starting it again.

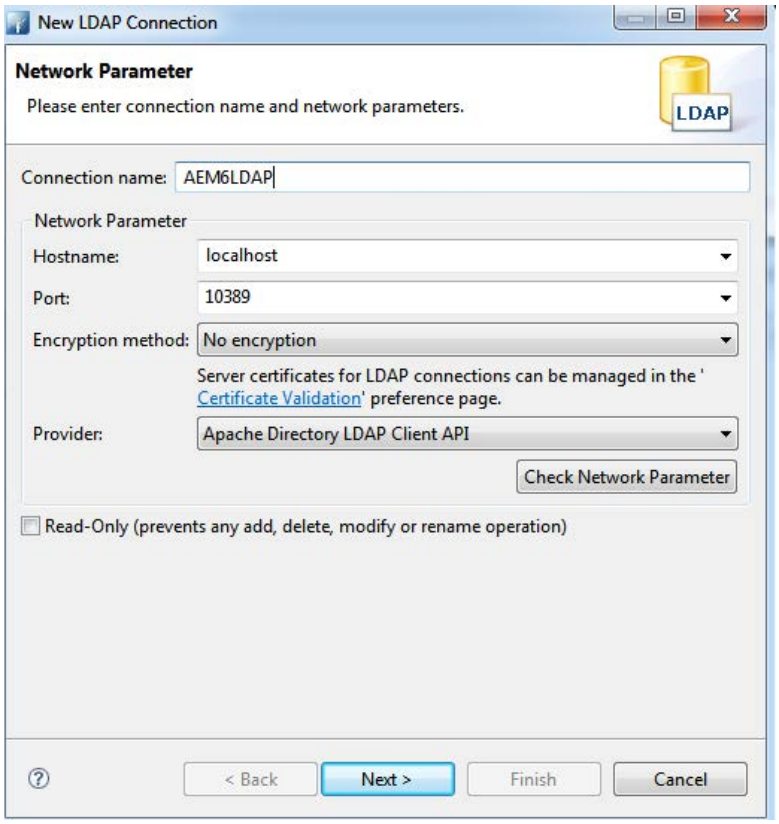


4. Stop the server, and then start it by clicking the green arrow.

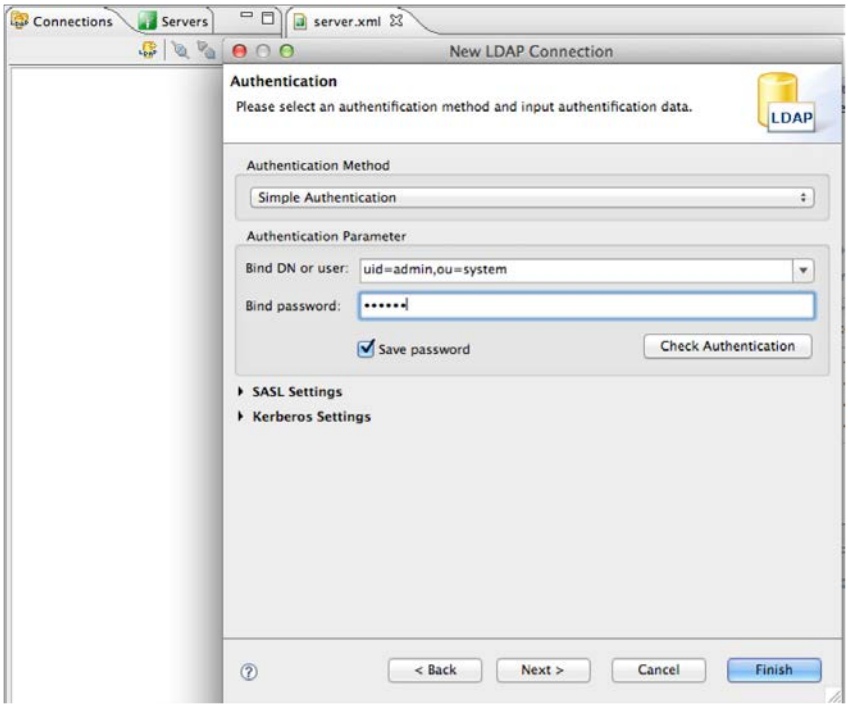


5. Right-click and select **Create a Connection**. On the **Server** tab, right-click, and click **Properties**, and then provide the details:
   Connection name: AEM6LDAP
   Hostname: localhost
   Port: 10389

6. Click **Next**.

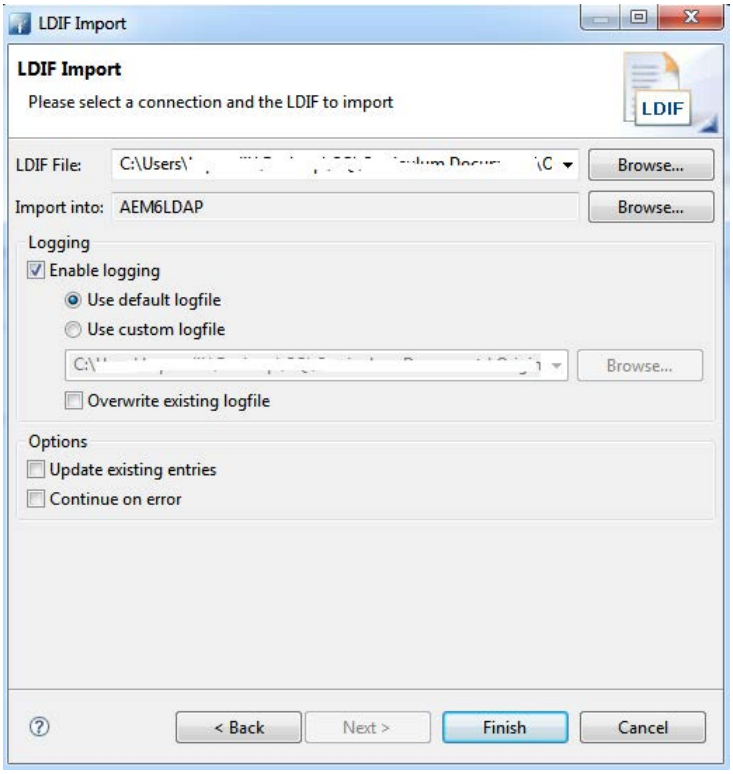


7. Fill in the authentication information:
   Bind DN or user: uid=admin, ou=system
   Bind password: secret



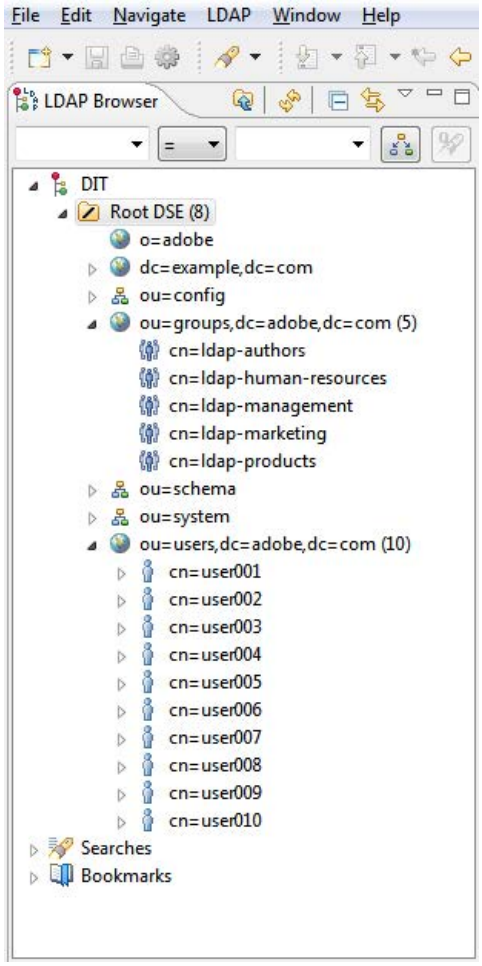
8. Click **Finish**.

9. In the **exercises/16.2 - import_ldap_users_groups** directory of the training memory stick, you will find the **users-groups-adobe.ldif** file. Import the ldif file into your LDAP server by right-clicking the **AEM6LDAP** connection and choosing **Import -> LDIF Import** from the menu.



10. Browse to the LDIF file and click **Finish**.

    Now, you have users and groups registered in the LDAP server.



Exercise 16.3
Configure AEM to Integrate With LDAP

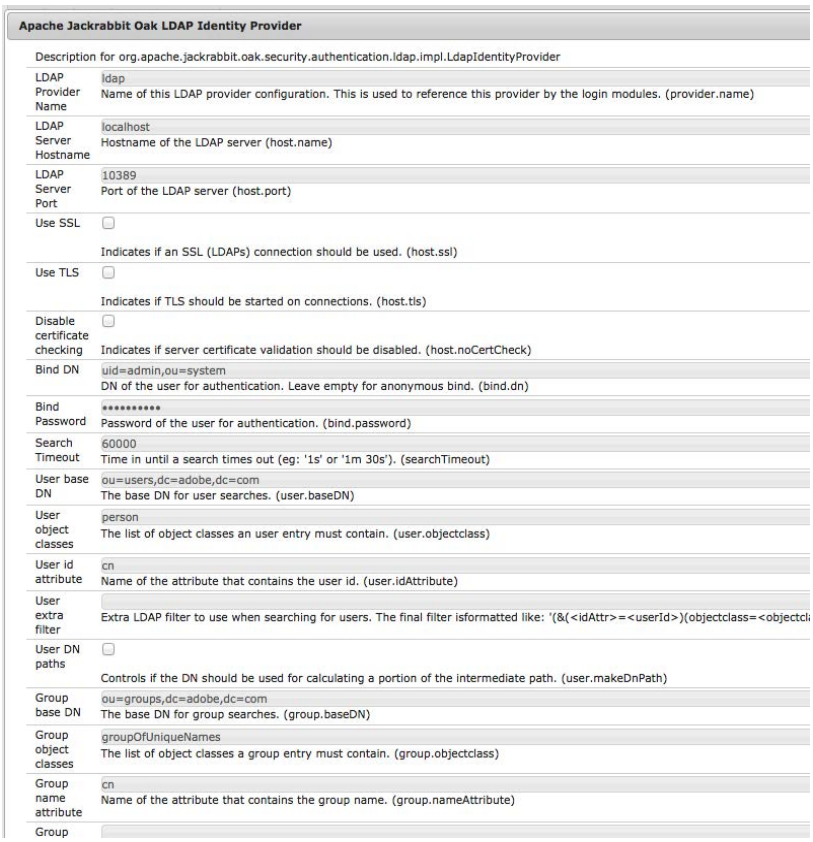
We will complete the three steps described at the beginning of the chapter by importing a package from **<USB path>\exercises\import_ldap_users_groups\ LDAP_ApacheDS-1.0.zip**. Importing this package will create the LDAP IDP, the synchronization handler, and the external login module with the following details.

1. Add a LDAP IDP

Find the **Apache Jackrabbit Oak LDAP Identity Provider** and create one configuration with the following values.

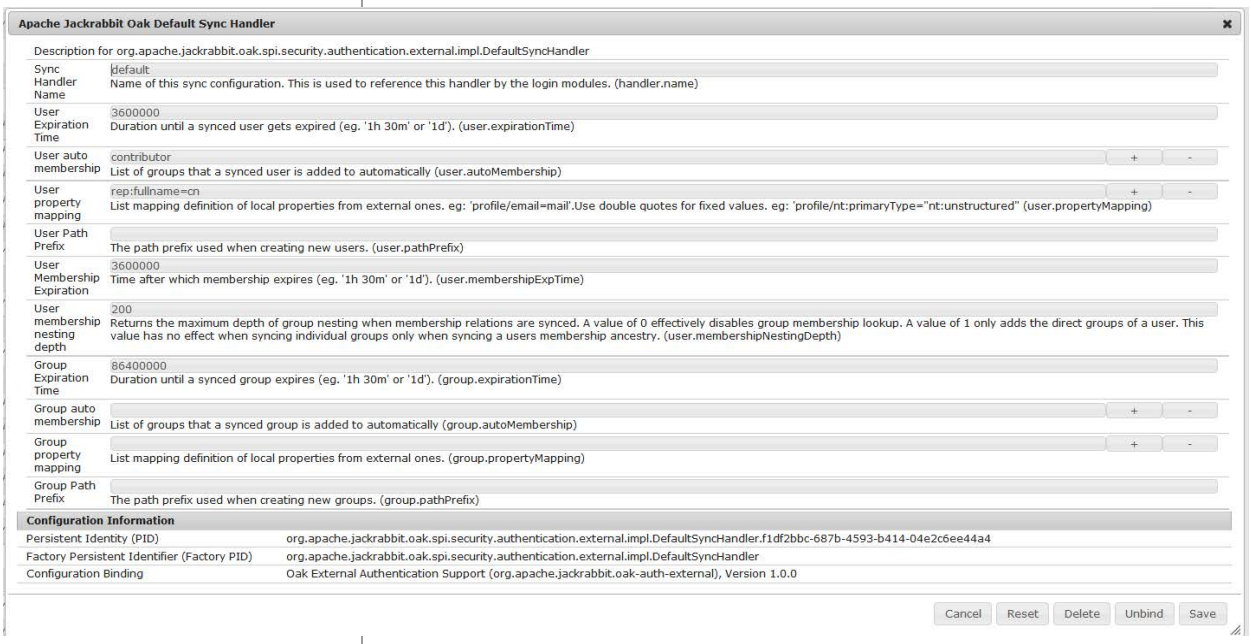
- LDAP Provider Name: ldap
- LDAP Server Hostname: 192.168.86.101 (Check and replace the IP of the machine and port number LDAP is running on.)
- LDAP Server Port: 10389
- Use SSL

- Use TLS
- Disable certificate checking
- Bind DN: uid=admin,ou=system
- Bind Password: secret
- Search Timeout: 60000
- User base DN: ou-users,ou-system
- User object classes: person
- User id attribute:uid
- User extra filter
- User DN paths
- Group base DN: ou=groups,ou=system
- Group object classes: groupOfUniqueNames
- Group name attribute: cn
- Group extra filter
- Group DN paths
- Group member attribute: uniquemember



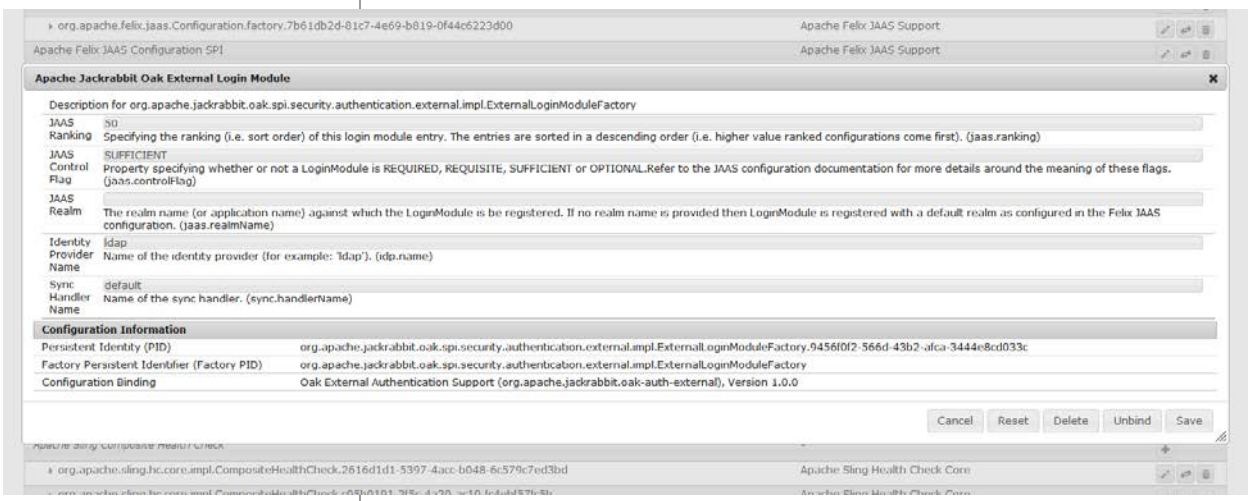
2. Add a Sync Handler

As shown in the screenshot below:



3. Add an External Login Module

As shown in the screenshot below:



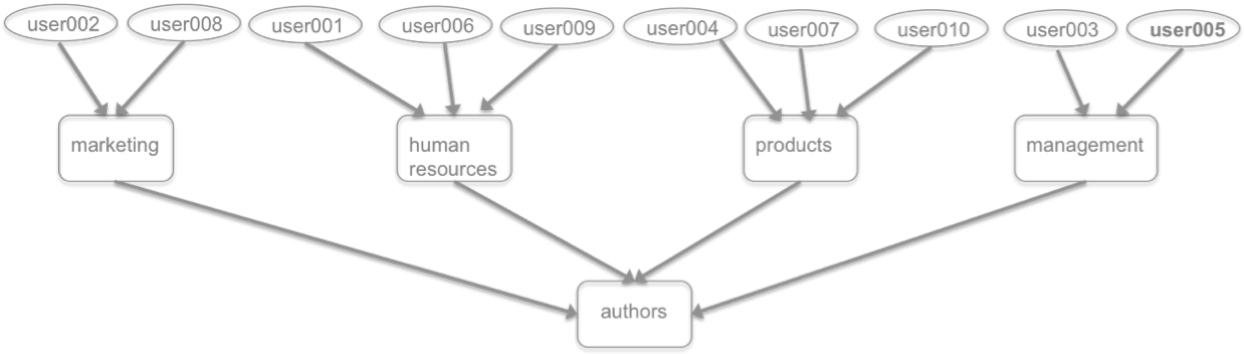
You can use the package provided under the chapter exercise **LDAP_ApacheDS-1.0.zip** to import all the settings in above three steps and test.

Exercise 16.4
Assign Rights to Users Imported From LDAP Into CRX

The provided LDAP example configuration file (ldif) contains five groups: ldap-authors, ldap-marketing, ldap-human-resources, ldap-products, and ldap-management. All four other groups are members of the ldap-authors group.

The users are distributed over department-specific groups; none of them is explicitly in the ldap-authors group because their specific groups are themselves members of the ldap-authors group. There are 10 users defined in the LDAP server: user001 – user010.

The password is the same for all users: '1234'. They are organized like this:



1. Log in to AEM with any of the test users (for example, u:**user001** p:**1234**). The user and group information are imported into AEM. However, you will receive a login error. At this point, the user does not have access to any resources in the repository as you have not yet defined permissions for that user.
2. Log in to AEM with administrator rights.
3. Refer to the previous exercise and assign appropriate page permissions to the **ldap-authors** group that was imported from LDAP.
4. Try logging in again as the LDAP user you previously imported, and then try another user (for example, user002).

NOTE: You may have to clear the browser cache, cookies, and active logins to successfully log in as admin.

Purge Users in LDAP

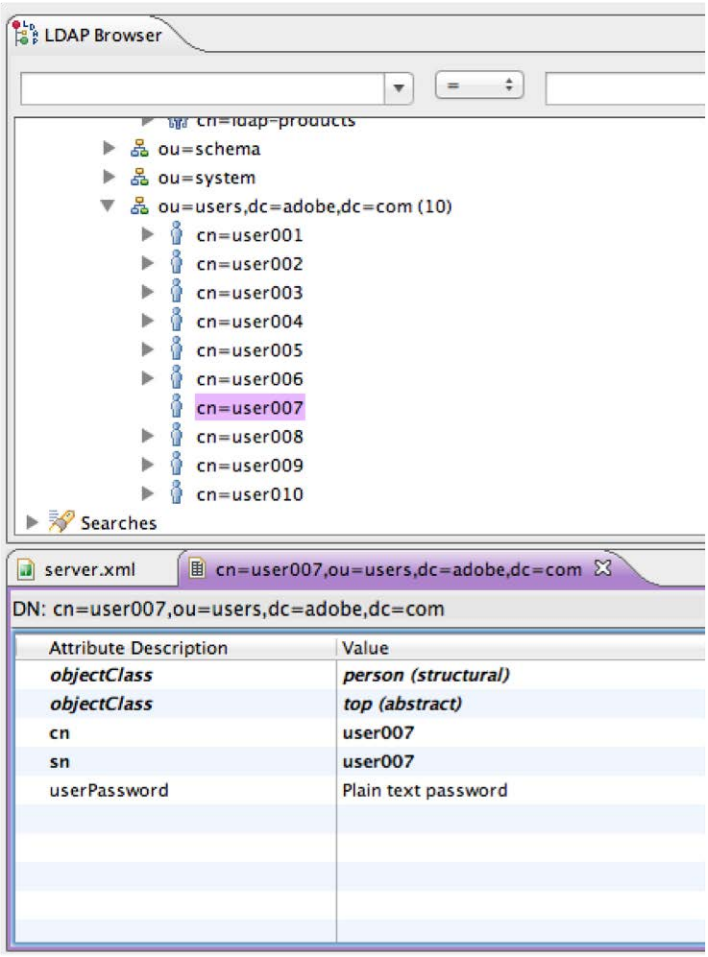
A common situation is to remove a user from the company-wide LDAP server. What happens in this case with the imported CRX users? The CRX status should be synchronized with LDAP; in other words, you need to remove those users. This can be done in CRX. You simply need to request the list of valid users from the connected LDAP server.

Exercise 16.5
Purge User007

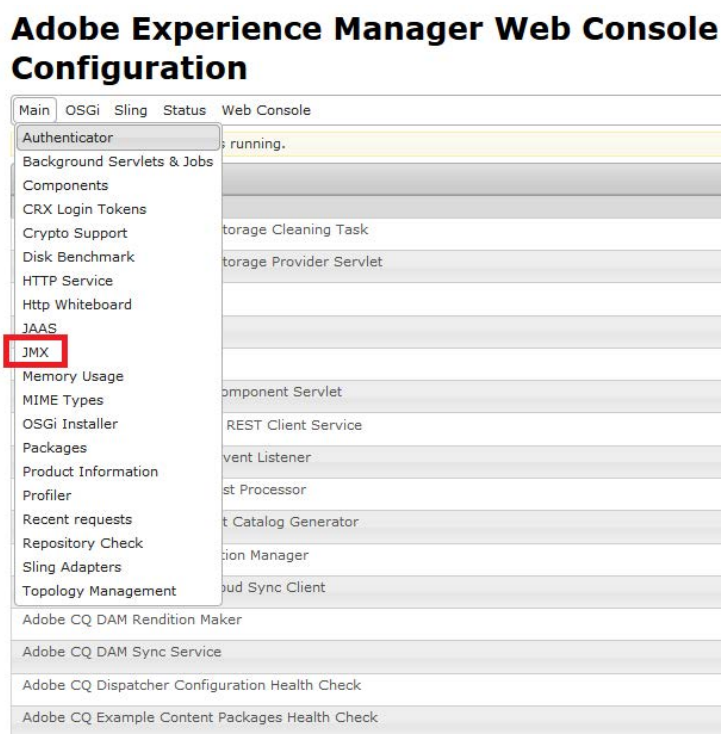
From your directory, you will remove user007 and synchronize CRX.

1. Access the LDAP Manager in the LDAP browser.
2. Make sure **user007** has previously logged-in AEM and has been added as a user in your AEM instance. If you cannot log in as that user, go to the LDAP browser and verify that the user exists and has a password attribute (user password) set.

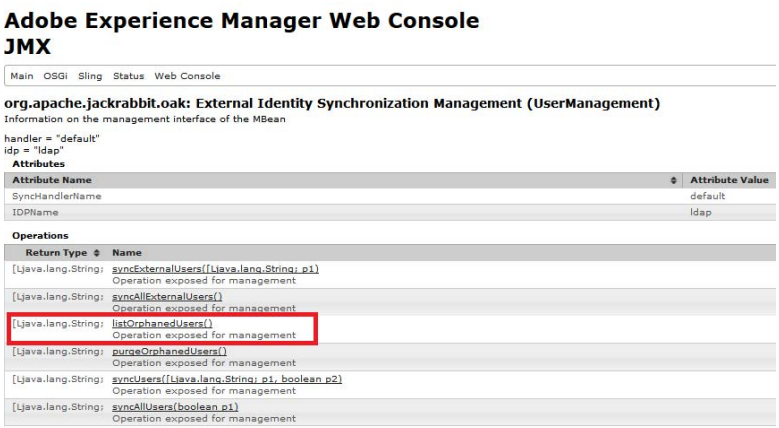
NOTE: To use all the repository functionality for LDAP synchronization, you will need to access the JMX console.



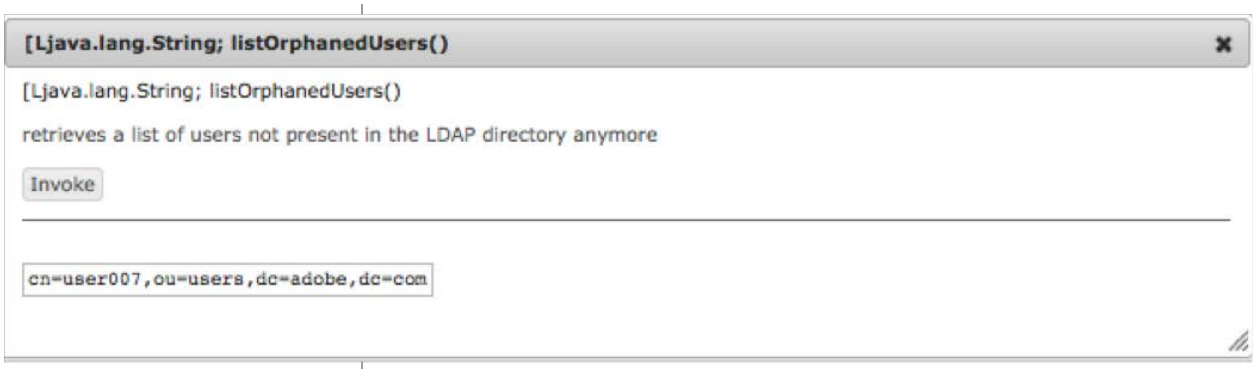
3. Select and remove **user007**.
4. Open the Adobe AEM Web Console, access the **JMX** tab and click the link, or go directly to http://localhost:4502/system/console/jmx.



5. In the JMX console, **select org.apache.jackrabbit.oak: External Identity Synchronization Management (UserManagement)**.



6. On the resulting LDAP tools page, click the **listOrphanedUsers()** link. In the popup, click the **Invoke** button. A list of users is displayed, which are no longer defined in LDAP but still existing in the CRX.



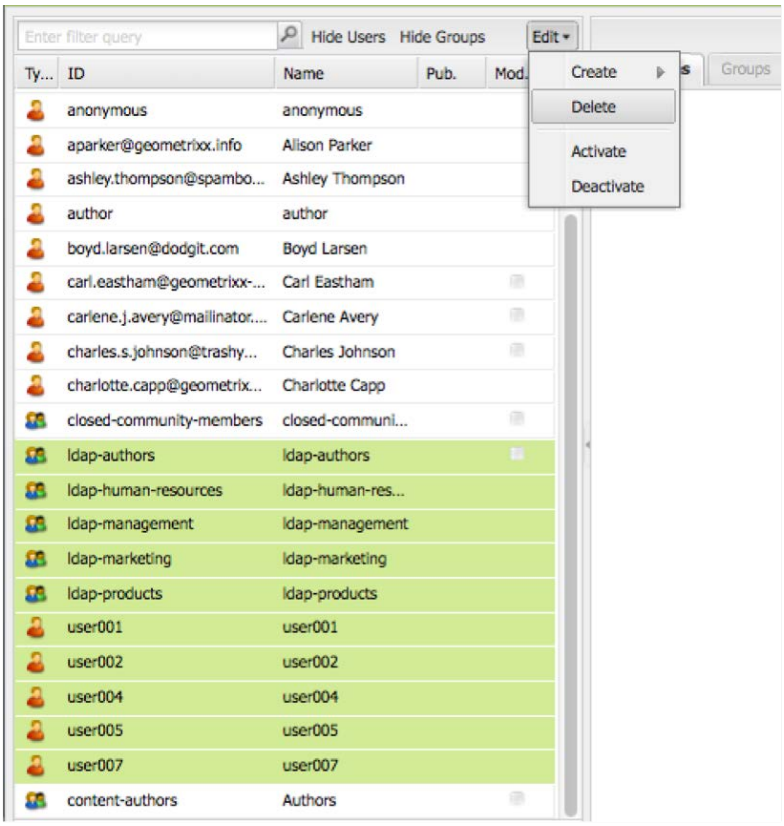
7. Close the **Orphaned Users** window and click the **purgeUsers()** link. In the emerging window, click the **Invoke** button.

Synchronize the CRX With LDAP

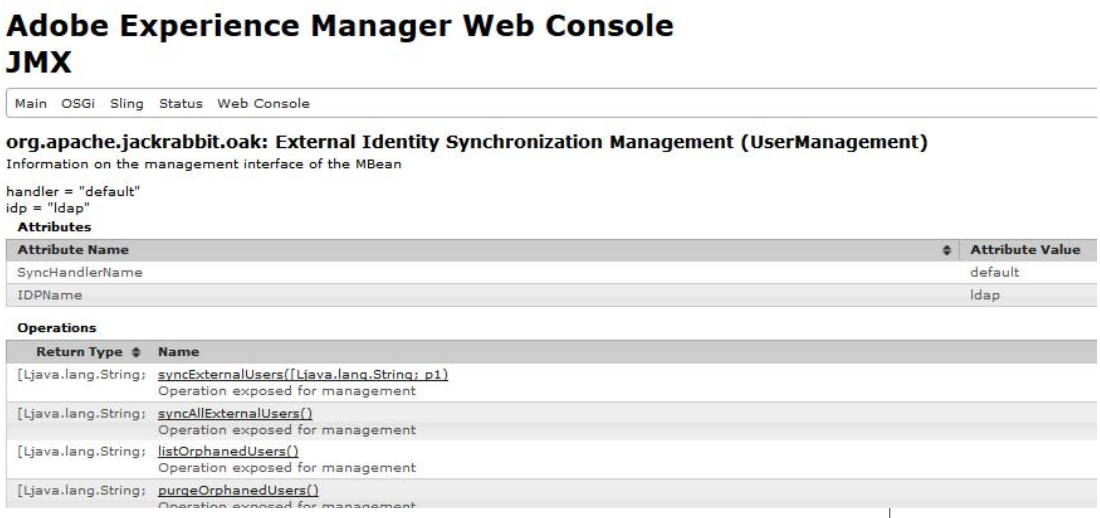
In one of the previous scenarios, it was demonstrated that CRX automatically imports a user after they are authorized by LDAP. The drawback was that during a first login attempt, because the user did not have any access control assigned, they received 404 messages. You, as SysAdmins, can easily avoid this situation and prevent users from getting a 404 message upon their first login attempt, if you synchronized the known groups that are supposed to have access to your site and give them the appropriate ACLs first.

Exercise 16.6
Synchronize the CRX With LDAP

For the purpose of this exercise, you will clean up your repository, removing any LDAP users and groups that you have imported from LDAP.



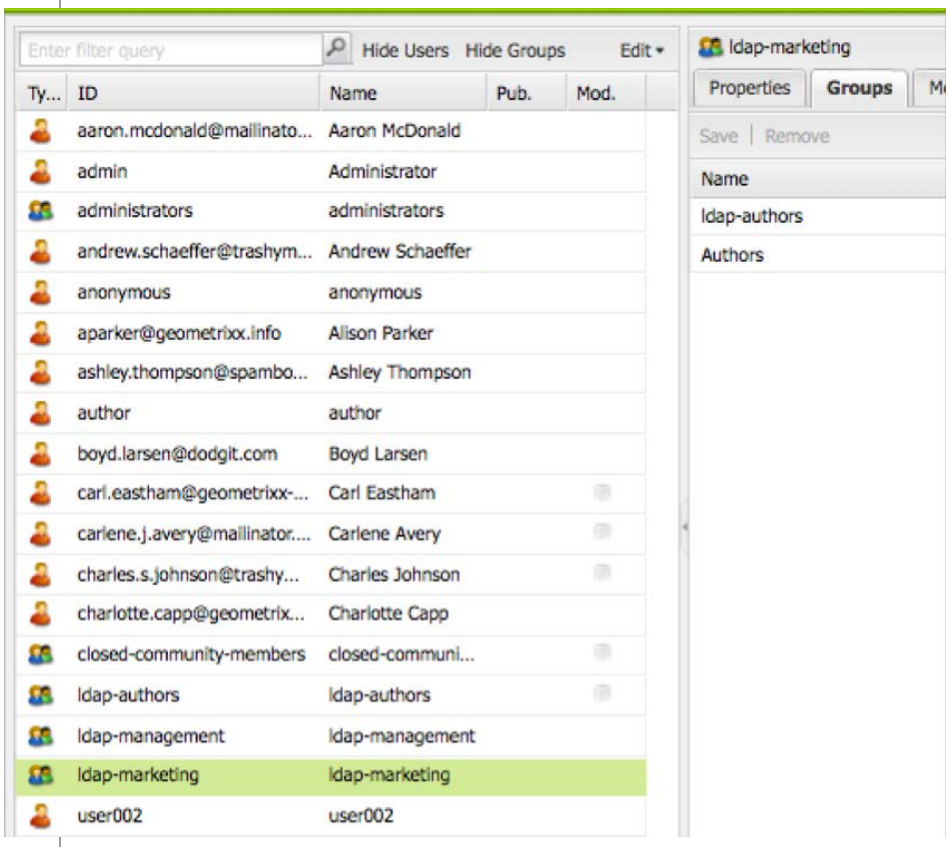
1. Open the Adobe AEM Web Console and access the **JMX** tab.
2. Click the **org.apache.jackrabbit.oak: External Identity Synchronization Management (UserManagement)** link. In the **attributes** section, you will see the LDAP server to which you are currently connected (**localhost**).



3. You can import a single user to CRX or several users listed as a text file. To synchronize only **user002**, you must click **syncUser(java.lang.String user)**

and enter: c**n=user002,ou=users,dc=adobe,dc=com** in the **user** field and click the **Invoke** button.

4. The user will be successfully imported to the CRX, but user002 still has no access permissions. Therefore, you must set the permissions before user002 attempts to log in.
   Go to the Users and Groups console and grant the user's group (ldap-marketing) with the desired access rights. We suggest making ldap-marketing a member of the content-authors or contributors group.



5. Try to log in to AEM with **user002**. It should work without problems. In case of difficulties, do not forget to clear the browser cache or close all browser windows and try again.

Additional Steps

You can also make sure that users are added to a group with appropriate permissions; for example, the contributors group, when they first log in. You can do this by making a change to the **ldap_login.conf** file.

6. Add the line `autocreate.user.membership="contributor"` to the **LDAPLoginModule**.
7. Restart AEM and try logging in as a different user.

Single Sign On

Single Sign On (SSO) allows a user to access multiple systems after providing authentication credentials (such as a user name and password) once. A separate system (known as the trusted authenticator) performs the authentication and provides Adobe Experience Manager with the user credentials. Adobe Experience Manager checks and enforces the access permissions for the user (that is, determines which resources the user is allowed to access).

The SSO Authentication Handler service (com.adobe.granite.auth.sso.impl.SsoAuthenticationHandler) processes the authentication results that the trusted authenticator provides. The SSO Authentication Handler searches for an SSO Identifier (ssid) as the value of a special attribute in the following locations in this order:

1. Request Headers
2. Cookies
3. Request Parameters

When a value is found, the search is finished and this value is used.
Configure the following two services to recognize the name of the attribute that stores the ssid:

- The login module
- The SSO authentication service

You must specify the same attribute name for both services. The attribute is included in SimpleCredentials that is provided to Repository.login. The value of the attribute is irrelevant and ignored; the mere presence of it is important and verified.

Configuring SSO

To configure SSO for an AEM instance, you need to configure the Adobe Granite SSO Authentication Handler, Various configuration properties are available as follows:

Path

Path for which this authentication handler is active. If this parameter is left empty , the authentication handler is disabled. For example, the path \ causes the authentication handler to be used for the entire repository.

Service Ranking

The OSGi Framework Service Ranking value is used to indicate the order used for calling this service. This is an int value where higher values designate higher precedence. The default value is 0.

Header Names

The name(s) of headers that might contain a user ID

Cookie Names

The name(s) of cookies that might contain a user ID

Parameter Names

The name(s) of request parameters that might provide the user ID

User Map

For selected users, the user name extracted from the HTTP request can be replaced with a different one in the credentials object. The mapping is defined here. If the user name admin appears on either side of the map, the mapping will be ignored. Be aware that the character '=' has to be escaped with a leading '\'.

Format

This indicates the format in which the user ID is provided. Use:

1. Basic, if the user ID is encoded in the HTTP Basic Authentication format
2. AsIs, if the user ID is provided in plain text or in regular expression, the applied value should also be used as plain text or as regular expression

When working with AEM there are several methods of managing the configuration settings for such services; see 'Configuring OSGi' for more details and the recommended practices.

NOTE:SSO is often used in conjunction with LDAP.

If you are also using the Dispatcher with the Microsoft IIS, then additional configuration will be required in:
1. The disp_iis.ini set:
servervariables=1 (forwards IIS server variables as request headers to the remote instance) and **replaceauthorization=1** (replaces any header named 'Authorization' other than 'Basic' with its 'Basic' equivalent)
2. IIS:
-Disable **Anonymous access**.
-Enable **Integrated Windows authentication**.

For example, for NTLM, set:

- **Path**: as required; for example, /
- **Header Names:** LOGON_USER
- **ID Format:** ^<DOMAIN>\\(.+)$
  where <DOMAIN> is replaced by your own domain name

For CoSign:

- **Path**: as required; for example, /
- **Header Names**: remote_user
- **ID Format**: AsIs

For SiteMinder:

- **Path**: as required; for example, /
- **Header Names**: remote_user
- **ID Format**: AsIs

Confirm that Single Sign On is working as required; including authorization.

CAUTION: Make sure that users cannot access AEM directly if SSO is configured.
By requiring users to go through a web server that runs your SSO system's agent, it is ensured that no user can directly send a header, cookie, or parameter that will lead the user to be trusted by AEM, as the agent will filter such information if sent from the outside.
Any user who can directly access your AEM instance without going through the web server will be able to act as any user by sending the header, cookie, or parameter if the names are known.
Also make sure that of headers, cookies, and request parameter names, you only configure the one that is required for your SSO setup.

Removing AEM Sign-Out Links

When using SSO, sign-in and sign-out are handled externally, so AEM's own sign-out links are no longer applicable and should be removed.

The sign-out link on the welcome screen can be removed using the following steps:

1. Overlay /libs/cq/core/components/welcome/welcome.jsp to /apps/cq/core/components/welcome/welcome.jsp.
2. Remove the following part from the jsp.

```
<a href="#" onclick="signout('<%= request.getContextPath()
%>');" class="signout"><%= i18n.get("sign out", "welcome
screen") %>
```

To remove the sign-out link that is available in the user's personal menu in the top-right corner, follow these steps:

3. Overlay /libs/cq/ui/widgets/source/widgets/UserInfo.js to /apps/cq/ui/widgets/source/widgets/UserInfo.js.
4. Remove the following part from the file:

```
menu.addMenuItem({
      "text":CQ.I18n.getMessage("Sign out"),
      "cls": "cq-userinfo-logout",
      "handler": this.logout
 });
 menu.addSeparator();
```

Optional Exercise 16.7
Configure LDAP Over SSL

AEM 6.1 can be configured to authenticate with LDAP over SSL by following the below procedure:

1. Select the **Use SSL** or **Use TLS** check boxes when configuring the LDAP IDP.
2. Configure the synchronization handler and the external login module according to your setup.
3. Install the SSL certificates in your Java VM if needed. This can be done by using keytool:

```
keytool -import -alias localCA -file <certificate location>
    -keystore <keystore location>
```

4. Test the connection to the LDAP server.

Optional Exercise 16.8
Create SSL Certificates

Self-signed certificates can be used when configuring AEM to authenticate with LDAP through SSL. Below is an example of a working procedure for generating certificates for use with AEM.

1. Make sure you have a SSL library installed and working. This procedure will use OpenSSL as an example.
2. Create a customized OpenSSL configuration (cnf) file. This can be done by copying the default openssl.cnf configuration file and customizing it. On UNIX systems, it is usually located at **/usr/lib/ssl/openssl.cnf**.
3. Proceed to creating the CA root key by running the following command in a terminal:

```
openssl genpkey -algorithm [public key algorithm] -out
    certificatefile.key -pkeyopt [public key algorithm option]
```

4. Next, create a new self-signed certificate:

```
openssl req -new -x509 -days [number of days for
    certification] -key certificatefile.key -out root-ca.crt
    -config CA/openssl.cnf
```

5. Inspect the newly generated certificate to make sure everything is in order:

```
openssl x509 -noout -text -in root-ca.crt
```

6. Make sure that all folders specified in the certificate configuration (.cnf) file exist. If not, create them.
7. Create a random seed, by running, for example:

```
openssl rand -out private/.rand 8192
```

8. Move the created .pem files to the locations configured in the .cnf file.
9. Finally, add the certificate to the Java keystore.

Enabling Debug Logging

Debug logging can be enabled for both the LDAP IDP and the external login module in order to troubleshoot connection issues.

In order to enable debug logging, you need to:

1. Go to the Web Management console.
2. Find **Apache Sling Logging Logger Configuration** and create two Loggers with the following options:
   - Log level: Debug
   - Log File logs/ldap.log
   - Message Pattern: {0,date,dd.MM.yyyy HH:mm:ss.SSS} *{4}* [{2}] {3} {5}
   - Logger: org.apache.jackrabbit.oak.security.authentication.ldap
   - Log level: Debug
   - Log File: logs/external.log
   - Message Pattern: {0,date,dd.MM.yyyy HH:mm:ss.SSS} *{4}* [{2}] {3} {5}
   - Logger: org.apache.jackrabbit.oak.security.authentication.external

NOTE: It is a known issue with LDAP and AEM that if you plan on using LDAP over SSL, make sure the certificates you are using are created without the Netscape comment option. If this option is enabled, authentication will fail with a SSL Handshake error.

Performance Tuning

Goal

The following instructions explain how to monitor page response times, in addition to finding slow page responses. This will allow you to create a more robust/scalable AEM application, even with limited hardware. To successfully complete and understand these instructions, you will need:

- A running AEM Author instance
- An Adobe-provided 'rlog.jar' tool (already installed with your AEM instance -<install dir>/crx-quickstart/opt/helpers)

What Performance Optimization Concepts Should I Consider?

A key issue is the time your website takes to respond to visitor requests. Although this value will vary for each request, an average target value can be defined. After this value is proven to be both achievable and maintainable, it can be used to monitor the performance of the website and indicate the development of potential problems.

The response times you will be aiming for will be different on the author and Publish instances, reflecting the different characteristics of the target audience:

Publish instance

This environment is used by authors entering and updating content, so it must cater for a small number of users who each generate a high number of performance intensive requests when updating content pages and the individual elements on those pages.

Publish instance

This environment contains content that you make available to your users, where the number of requests is even greater and the speed is just as vital. As the nature of the requests is less dynamic, additional performance-enhancing mechanisms can be leveraged, such that the content is cached or load balancing is applied.

Performance Optimization Methodology

A performance optimization methodology for AEM projects can be summed up to five simple rules that can be followed to avoid performance issues from the start. These rules, to a large degree, apply to web projects in general, and are relevant to project managers and system administrators to ensure that their projects will not face performance challenges when launch time comes.

Planning for Optimization

Around 10% of the project effort should be planned for the performance optimization phase. Of course, the actual performance optimization requirements will depend on the level of complexity of a project and the experience of the development team. While your project may ultimately not require all of the allocated time, it is good practice to always plan for performance optimization in that suggested range.

Whenever possible, a project should first be soft-launched to a limited audience to gather real-life experience and perform further optimizations, without the additional pressure that follows a full announcement.

When you are 'live', performance optimization is not over. This is the point in time when you experience the 'real' load on your system. It is important to plan for additional adjustments after the launch.

Because your system load changes and the performance profiles of your system shift over time, a performance 'tune-up' or 'health check' should be scheduled at 6-to-12-month intervals.

Simulate Reality

If you go live with a website and you find out after the launch that you run into performance issues, there is only one reason for that—your load and performance tests did not simulate reality close enough.

Simulating reality is difficult, and how much effort you will reasonably want to invest into getting 'real' depends on the nature of your project. 'Real' means not just 'real code' and 'real traffic', but also 'real content', especially regarding content size and structure. Keep in mind that your templates may behave completely different depending on the size and structure of the repository.

Establish Solid Goals

The importance of properly establishing performance goals is not to be underestimated. Often, when people are focused on specific performance goals, it is hard to change these goals afterward, even if they are based on wild assumptions.

Establishing good, solid performance goals is really one of the trickiest areas. It is often best to collect real-life logs and benchmarks from a comparable website (for example, the new website's predecessor).

Stay Relevant

It is important to optimize one bottleneck at a time. If you do things in parallel without validating the impact of the one optimization, you will lose track of which optimization measure actually helped.

Agile Iteration Cycles

Performance tuning is an iterative process that involves measuring, analysis, optimization, and validation until the goal is reached. To properly consider this aspect, implement an agile validation process in the optimization phase rather than a more heavyweight testing process after each iteration.

This largely means that the administrator implementing the optimization should have a quick way to tell if the optimization has already reached the goal, which is valuable information, because when the goal is reached, optimization is over.

Basic Performance Guidelines

Generally, keep your uncached HTML requests to less than 100ms. More specifically, the following may serve as a guideline:

- 70% of the requests for pages should be responded to in less than 100ms.
- 25% of the requests for pages should have a response within 100ms-300ms.
- 4% of the requests for pages should have a response within 300ms-500ms.
- 1% of the requests for pages should have a response within 500ms-1000ms.
- No pages should respond slower than 1 second.

These numbers assume the following conditions:

- Measured on publish (no authoring environment and/or CFC overhead)
- Measured on the server (no network overhead)
- Not cached (no AEM-output cache, no Dispatcher cache)
- Only for complex items with many dependencies (HTML, JS, PDF)
- No other load on the system

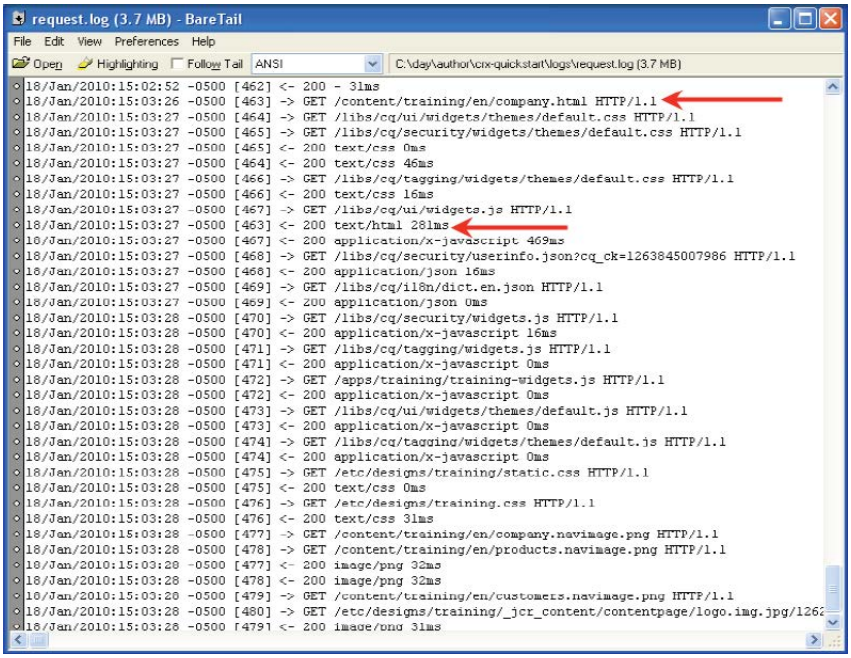
There are a certain number of issues that frequently contribute to performance issues, which mainly revolve around

a. Dispatcher caching inefficiency
b. The use of queries in normal display templates.

JVM and OS level tuning usually do not lead to big leaps in performance and should therefore be performed at the very tail end of the optimization cycle.
Your best friends during a usual performance optimization exercise are the request.log, component- based timing, and a Java profiler.

Exercise 17.1
Monitor Page Response

1. Navigate to, and open the **request.log** file located at **<cq-install-dir>/crx-quickstart/logs**.
2. Request a page in Author that utilizes your training template and components.
   - For example, */content/training-site/en/company*
3. Review the response times directly related to the previous step's request.
   - A page request of /content/training-site/en/company



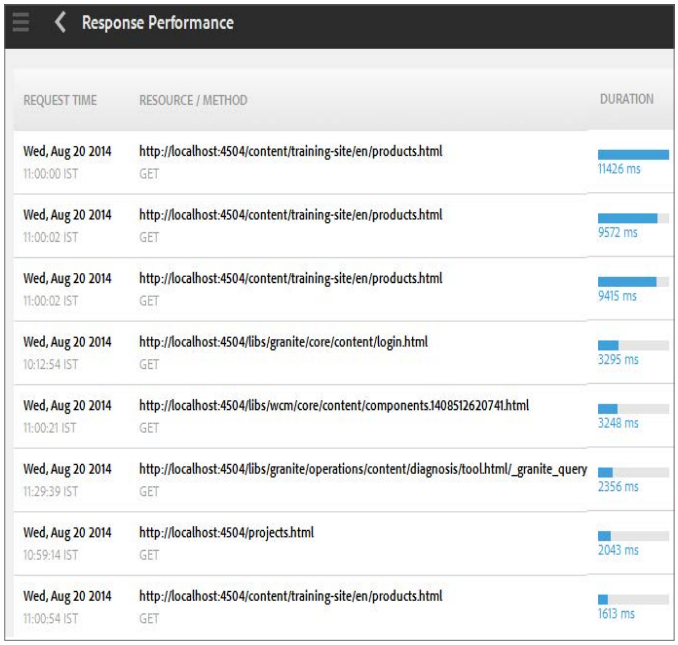
request.log Response Time

You have now successfully reviewed the response time of a page in AEM using request.log. Again, this will aid your development in being able to monitor response times of pages that implement custom templates and components, and comparing said time to your project goals.

Exercise 17.2
Find the Response Performance

1. Log in to AEM Sites.
   http://localhost:4502

2. Go to **Tools** > **Operations** > **Dashboard** > **Diagnosis** > **Request Performance.** The Request Performance page appears. The page displays requests in the order of duration taken.

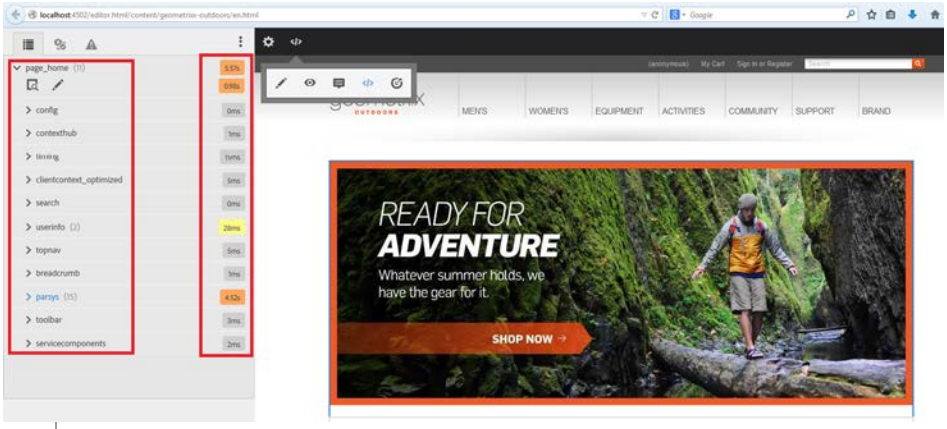


You can use this information to further investigate the cause of the long response.
You have now successfully found and displayed long lasting requests/responses in AEM. Again, this is just one of many tools to help you meet your project's performance goals.

Exercise 17.3
Monitor Component-based Timing

1. Navigate and open the page you want to monitor. We are checking **http://localhost:4502/editor.html/content/geometrixx-outdoors/en.html** in this section.

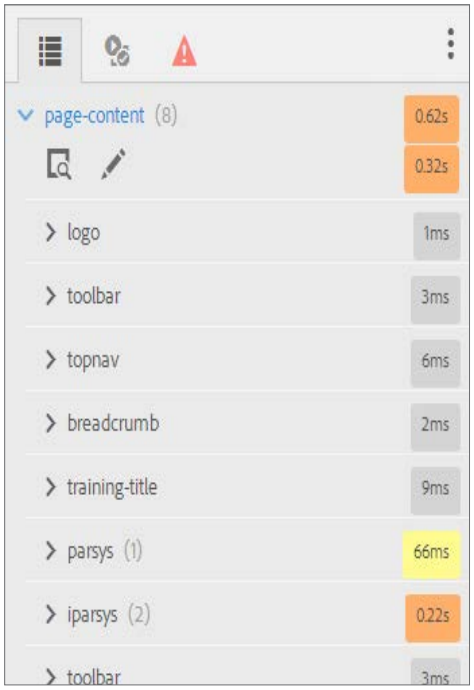
NOTE**:** Though this is clearly the response time of a page while in author, it is a good practice to be able to identify where response times are located (request.log) and how to identify a specific page request/response in the log files itself. Ideally, this test would occur on the Publish Instance for a better idea of its expected behavior in production.



NOTE: Though this is clearly the response time of a page while in Author, it is good practice to be able to identify where response times are located (request.log) and how to identify a specific page request/response in the log file itself. Ideally, this test would occur on the Publish Instance for a better idea of its expected behavior in production.

2. Observe that the page components and the time they took to load is highlighted in the screenshot above.

In the Developer mode of the page, you can view the time taken by the components to load The Components tab displays a component tree that provides you with the server-side computation time for each component.



Congratulations! You successfully monitored component-based timing using the foundation-timing component. Again, this will aid your development in being able to monitor the response times of specific components within the context of an actual page/template request. Next, you will focus on how to find long-lasting requests.

Security Checklists

CRX Security Checklist

Installation Settings

Determine the Security Level: Make sure that there is no 'allow Access Control' entry on the root node (/) for the principal 'everyone'. This would allow everyone to read/write content.

Switch from Low to High Security: To switch from low to high security, you need to remove all access control policies for the principal everyone on the root node. This will deny default read/write access to content available to any logged-in user. After this, you will be able to configure authorization for your repository groups and users according to your needs.

Passwords

Change the CRX admin password before you use CRX productively. It is recommended that you change the default passwords. The default password for the admin account is admin.

The password protects the Anonymous account.

Issues with Cross-Site Request Forgery

To address known security issues with Cross-Site Request Forgery (CSRF) in CRX WebDAV and Apache Sling, you need to configure the Referrer filter.

The referrer filter service is an OSGi service that allows you to configure:

- That which http methods should be filtered
- That whether an empty referrer header is allowed
- That a white list of servers to be allowed in addition to the server host

By default, all variations of localhost and the current host names the server are bound to be in the white list.

File System Security

Prohibit Illegitimate Write Access to CRX Installation: Write access to files in a CRX installation, or its backups must only be allowed to a users whose role in CRX is that of

an admin. Write access to the installation folders allows users to reconfigure CRX through OSGi configuration files, which allows immediate rights elevation to admin.
Prohibit Illegitimate Read Access to Log Files: Log files can potentially disclose information that reveals attack vectors. As such, access to read log files must only be granted to trusted entities.

Securing HTTP Communication

As with any application available on the Internet, any confidential information must be secured on a transport level.
The need for this will vary depending on the content stored in your system.
To ensure that the system cannot be compromised, the following should be secured through SSL for any CRX system, irrespective of the content in the system.

- Access to the OSGi console
- Access by the admin user
- Access to user information
- Inter-system communication if it goes over the public Internet

Sanity Checks Prior to Go-Live

Various areas are impacted and you should review all in detail. The following list is meant to provide an introductory overview to see, which can be used to help protect your implementation:

- Change Default Passwords
  An out-of-the-box installation of AEM includes various accounts to enable you to administer and use the instance.
  In particular, we strongly recommend that after installation, you change the passwords for the privileged (admin) account(s).
- Uninstall Example Content and Users
  All example content and users should be uninstalled completely on a productive system before making it publicly accessible.
- Disable CRXDE Support
  CRXDE Lite support should be disabled on a productive system before making it publicly accessible.
- Configure Replication and Transport Users
  Create users with specific, restricted access rights for building replication content (Author instance) and for receiving content (Publish instances) instead of using the admin user.
- Disable WebDAV
  CRX and AEM come with WebDAV support that lets you display and edit the repository content. Setting up WebDAV gives you direct access to the content repository through your desktop.
  WebDAV should be disabled on the Publish instance.
- Use the Latest Version of Dispatcher

- Restrict Access Through the Dispatcher
  The Dispatcher filter can be used to allow or deny external access to specific areas of AEM. To protect your instance, you should configure the Dispatcher to restrict external access as far as possible.
- Protect Against Cross-Site Scripting (XSS)
  Cross-Site Scripting (XSS) allows attackers to inject code into webpages viewed by other users. This security vulnerability can be exploited by malicious web users to bypass access controls.
  AEM applies the principle of filtering all user-supplied content upon output. Additionally, a web application firewall can be configured to add protection.
- Preventing Denial of Service (DoS) Attack
  A Denial of Service (DoS) attack is an attempt to make a computer resource unavailable to its intended users. This is often done by overloading the resource. There are a few methods that can be used with AEM to help prevent such attacks.
- Default Access to User Profile(s) Is Everyone
  By default, everyone (the built-in group) has read access to all user profiles. If such access is not appropriate for your installation, you can change these default settings.
- Issues With CSRF
  This is a security issue from the CRX Security Checklist, which is also appropriate to AEM. To address known security issues with CSRF in CRX WebDAV and Apache Sling, you can configure the Referrer filter.
- Disable the CQ WCM Debug Filter on Production Systems
  This is useful when developing as it allows the use of suffixes, but should be disabled on a production instance to ensure performance and security.
- Clickjacking
  Configuring your web server can help prevent clickjacking.
- Access to Cloud Service Information
  Reviewing whether the default security on Cloud Service Information matches your requirements is recommended.
- OSGi Settings
  Changing some OSGi settings on your publish instances can help you secure the information.

In one of the previous exercises, you learned how to protect your AEM instance(s) from unwanted visitors. Now let us make your system more secure. Out of the box, AEM is installed to accept any kind of requests; it is your responsibility to control what kind of requests you want to provide to visitors and refuse those requests, which can endanger your data. The Publish instances are typically exposed directly to visitors' access. You performed some restrictions to interfaces like /crx, /admin, and so on as you configured the AEM Dispatcher. In a default environment, WebDAV access is allowed by AEM and cannot be filtered out by the AEM Dispatcher because WebDAV is based on

HTTP and requests to/is generally allowed. You want to restrict only WebDAV access to Publish1.

Exercise 18.1
Disable WebDAV and Debug Mode

1. Open the Adobe AEM Web Console on the Publish Instance. Click the **OSGi** drop-down menu and select **Configuration**. Navigate to **Day CQ WCM Debug** Filter.
2. Clear the **Enabled** check box.

NOTE: For the purposes of expediency, we are using the Console to set this value. In production, you will create a sling:Osgi node of the name com.day.cq.wcm.core.impl.WCMDebugFiilter and set properties on that node. Then, you will be able to create different configurations of the debug filter for different environments and distribute them automatically through CRX content packages.

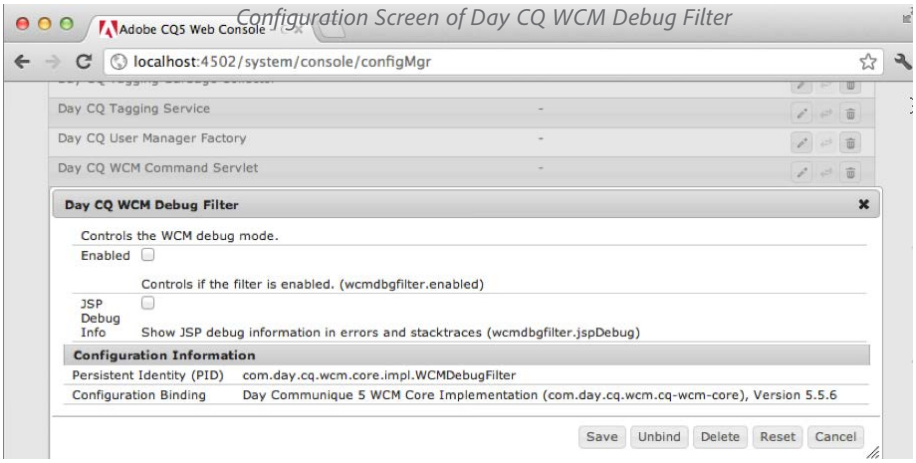
WebDAV Bundle After It Is Stopped

Test your modification; try to establish a WebDAV connection to this instance. As expected, you should not be allowed to connect, while an attempt to connect on the unprotected Publish2 should work.

Disable Debug Mode

Developers can receive detailed information about scripts running in the background to render the page content as desired. You can test it by appending the parameter '?debug=layout' to any URL, for example, http://localhost:4502/content/geometrixx/en.html?debug=layout. The page is now displayed with other component details. As you can imagine, we want to prevent visitors to view the internals of our installation. To disable this 'debug mode' on the Publish Instance(s):

1. Open the OSGi console **Apache Felix** on the Publish Instance and navigate to the configuration module **CQ Day CQ WCM Debug Filter**.
2. Clear the **Enabled** check box.



Configuration Screen of Day CQ WCM Debug Filter

3. Save your settings.

Test your modification. You have two Publish Instances: one protected and one vulnerable. See and understand the difference.

Production-Ready Lockdown Package in AEM 6.1

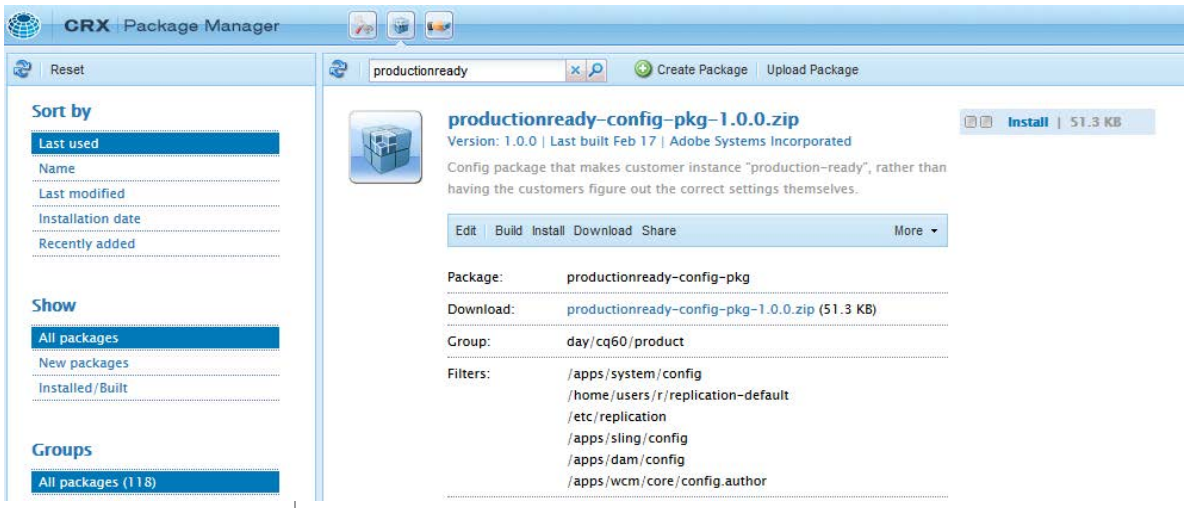
AEM 6.1 provides two packages out of the box that can be installed and used to lock down the security threats to your production system.

Go to Package Manager at http://localhost:4502/crx/packmgr/index.jsp. In the search packages field, enter **productionready**. You will see the following packages:

- productionready-config-pkg-1.0.0.zip
- productionready-hc-pkg-1.0.1.zip

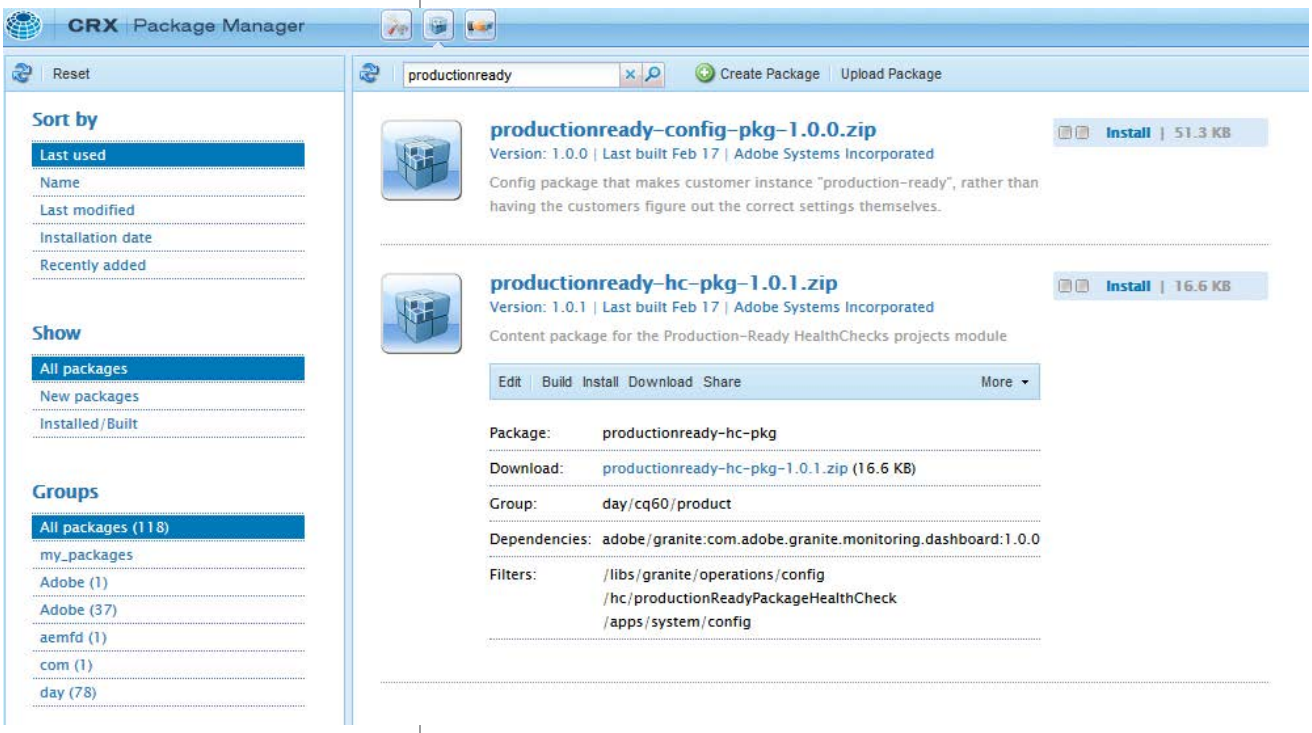
You will observe that productionready-config-pkg-1.0.0.zip contains the configurations for various parts of AEM as:

- /apps/system/config
- /home/users/r/replication-default
- /etc/replication
- /apps/sling/config
- /apps/dam/config
- /apps/wcm/core/config.author



Similarly, productionready-hc-pkg-1.0.1.zip has the health check report for your system under:

- /libs/granite/operations/config/hc/productionReadyPackageHealthCheck
- /apps/system/config



Existing System Console and New Dashboard

Operations Dashboard

The all-new Operations Dashboard in AEM 6.1 helps system operators to monitor AEM system health at a glance. It also provides auto-generated diagnosis information on relevant aspects of AEM and allows configuring and running of self-contained maintenance automation to reduce project operations and support cases significantly. The Operations Dashboard can be extended with custom health checks and maintenance tasks. Further, Operations Dashboard data can be accessed from external monitoring tools via JMX.

The dashboard shows the status of various parts of AEM and highlights any problem areas. These dashboards can also show when maintenance tasks must be run.
The Operations Dashboard:

- is a one-click system status to help operations departments gain efficiency.
- provides system health overview in a single, centralized place.
- dramatically reduces time to find, analyze, and fix issues.
- provides self-contained maintenance automation that helps reduce project operations costs significantly.

The dashboard can be accessed by navigating to **Tools**-> **Operations**-> **Dashboard**
The following options are available under Dashboard:

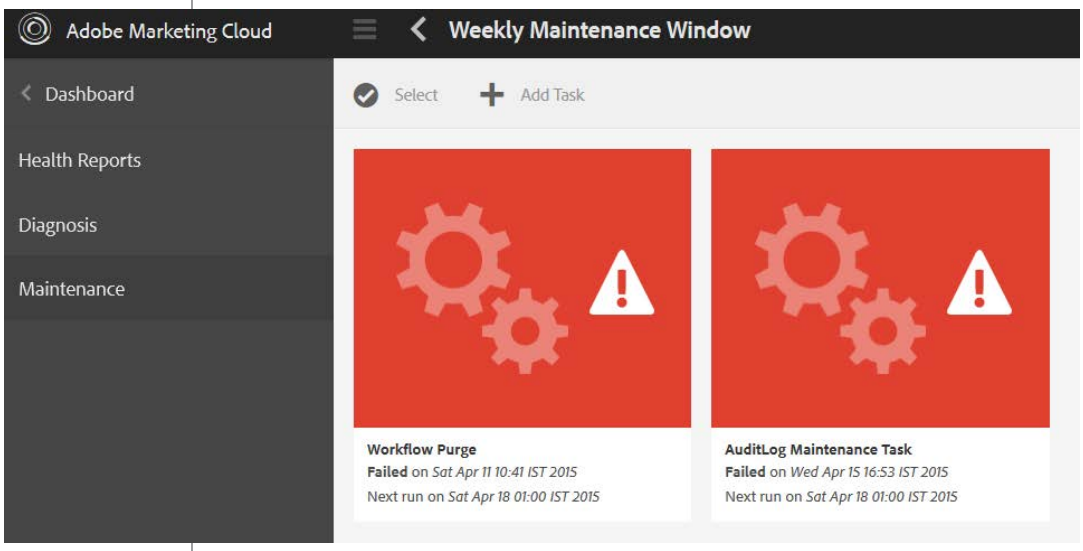
- Health Reports
- Diagnosis Tools
- Automated Maintenance

Audit Log Maintenance in AEM 6.1

AEM events that qualify for audit logging generate much archived data. This data can quickly grow over time due to replications, asset uploads, and other system activities.

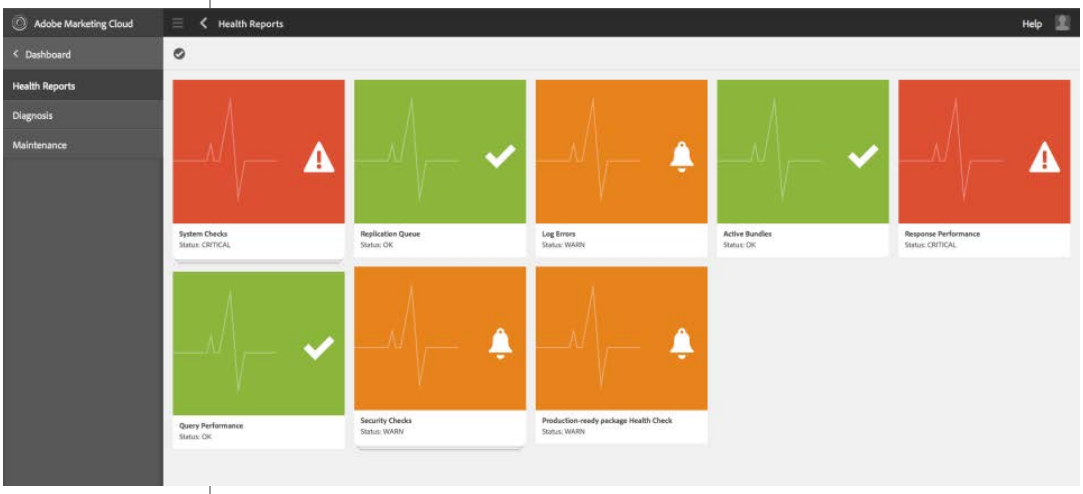
The Audit Log Maintenance includes several parts of functionality that enables the ability to automate audit log maintenance under specific policies.

It is implemented as a configurable weekly maintenance task and is accessible via the Operations Dashboard monitoring console.

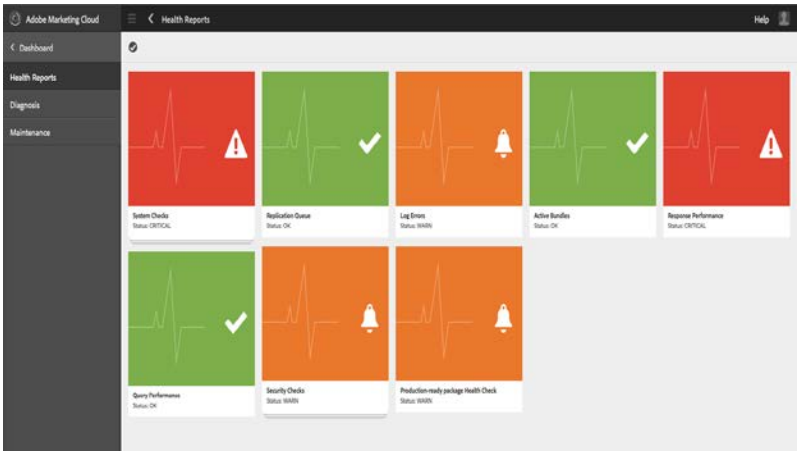


Exercise 19.1
Work With System Checks

1. The first step is to consult existing cards and configurations concerning the running operations:
   a. Go to **Tools -> Operations -> Dashboard -> Health Reports** in the AEM Touch Interface or open: http://localhost:4502/libs/granite/operations/content/healthreports.html

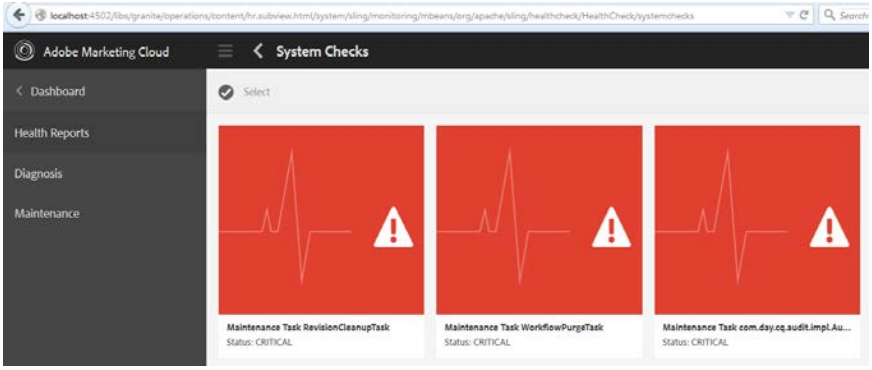


Notice that the **System Checks** card is in a **CRITICAL** state:

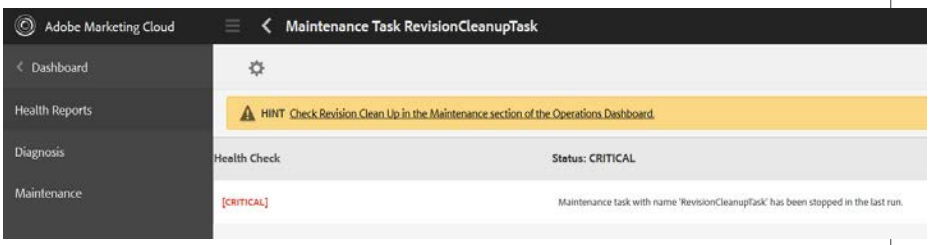


As the **System Checks** card is in the **CRITICAL** state, you can consult the next level of information and begin to resolve the issue.

Clicking directly on the card will displays the next level of information with three cards in critical state:

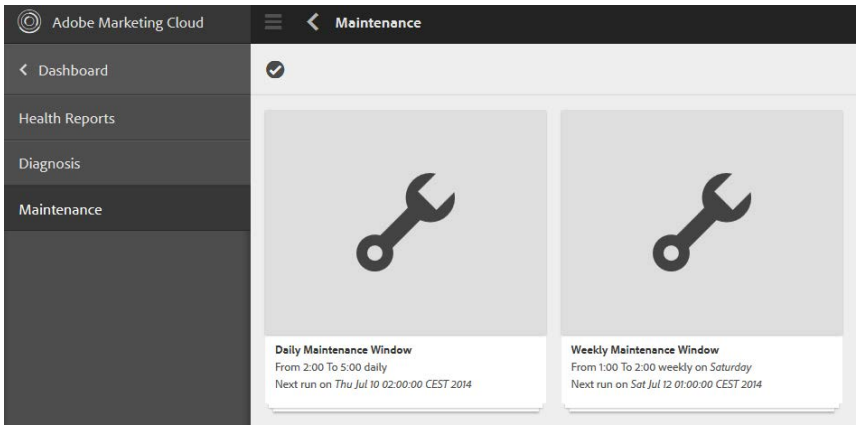


b. Click the **RevisionCleanUpTask** maintenance task card to see the next level of information and the 'hint' on how to fix it:



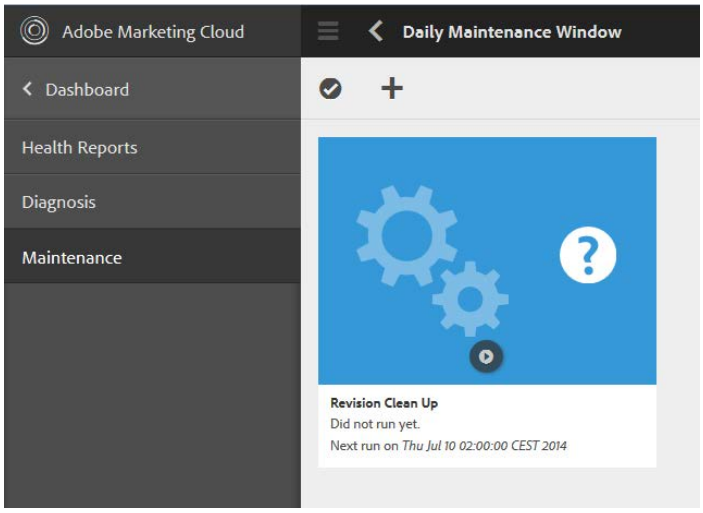
2. click the **HINT** link shown above, which leads you to the Maintenance Tasks page on the Operations dashboard:

NOTE: Alternatively, you can click on the 'Info' button (on mouseover) you will be brought to a page describing all the errors, giving you two hints—one hint for the 'RevisionCleanupTask' and the other for the 'WorkflowPurgeTask.'



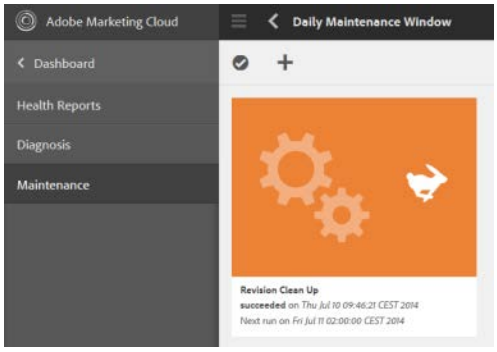
In the **Maintenance** page, you can perform three tasks:

a. The first maintenance task will be to run **Revision Clean Up** (RevisionCleanUpTask) found under **Daily Maintenance Window**:

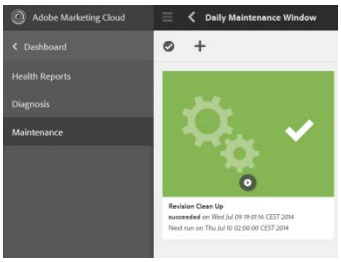


To launch the task, tap the card while using a tablet or click the **Play** button that appears when you hover the cursor over the card. As it runs the task, you will notice how the associated card first turns:

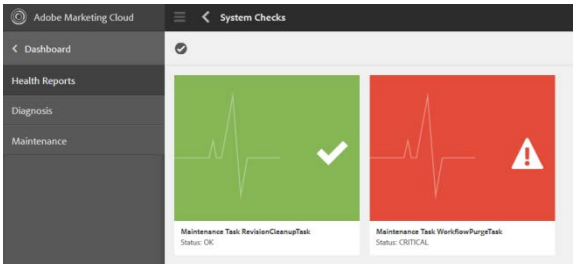
i. Orange while running:



ii. Green on completion of the **Daily Maintenance Window** tasks:

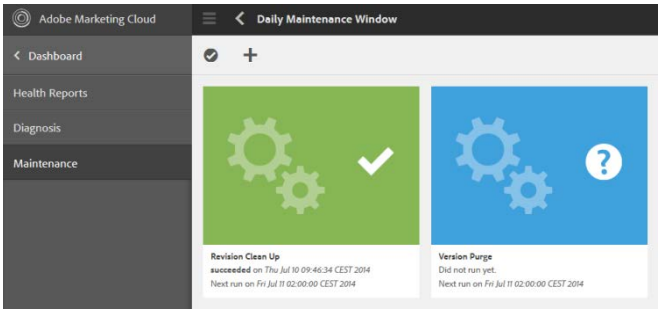


You can also see that the card also turns green on the **System Checks** page once all three tasks are completed:



b. The second maintenance task is to add and run a new **Version Purge** task in the **Daily Maintenance Window**:

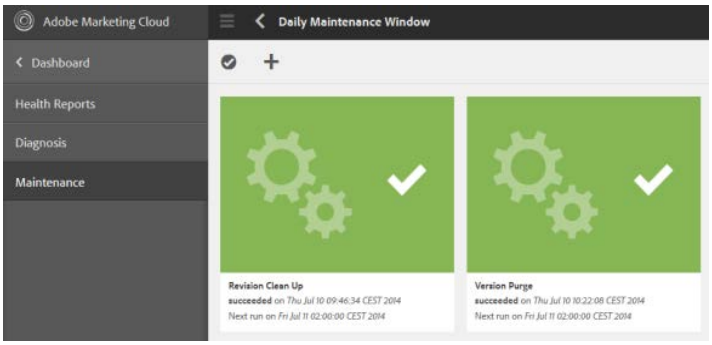
i. Click the **+** button within the **Daily Maintenance Window** to add a new **Version Purge** task. A new blue card will appears.



ii. Before you run the new task, go to the **System Checks** page and notice how a fourth card appears.

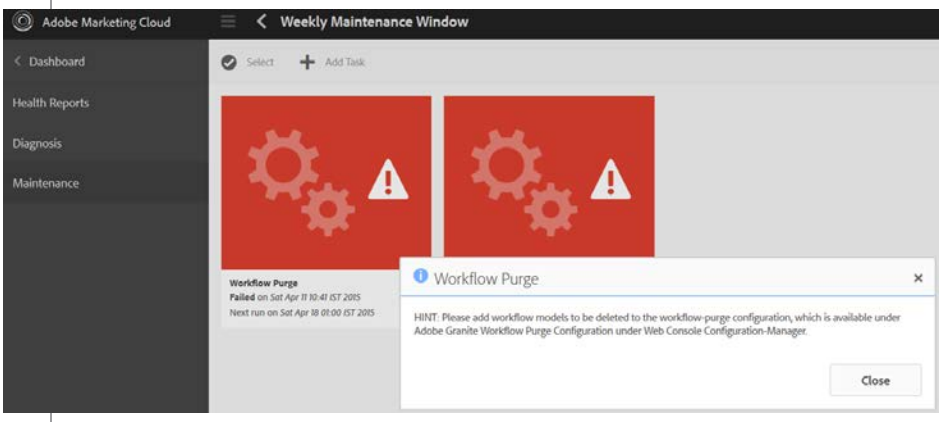
This fourth card appears in the **CRITICAL** state because it has not yet run. Return to the **Daily Maintenance Window** and run the task. The card turns orange while the task is running, and then green when the

task is finished. The task card will also change to green in the **System Checks** window.

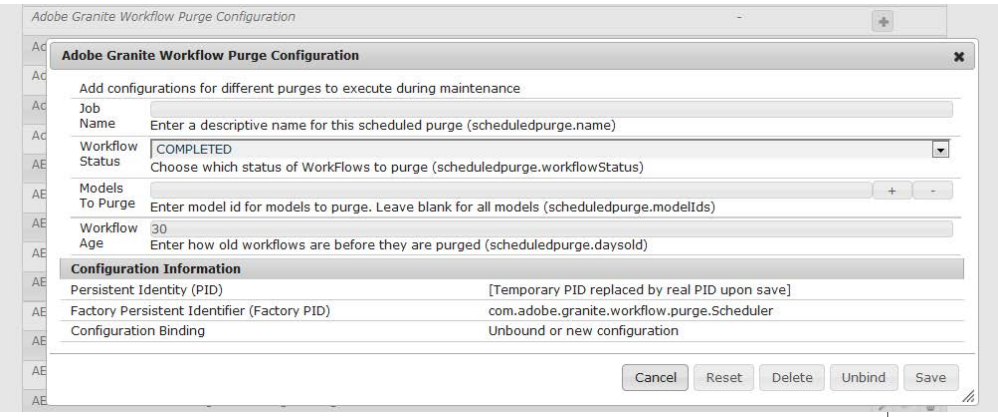


c. The third thing you will accomplish in the **Daily Maintenance Window** is to add and run a configuration for the **Workflow Purge Task**.

   i. First, consult the 'hint' in the **Weekly Maintenance Window** about configuring the purge task, which states: 'HINT: Please add workflow models to be deleted to the workflow-purge configuration, which is available under Adobe Granite Workflow Purge Configuration under Web Console Configuration-Manager.



   ii. Go to the **Web Console** to find the workflow purge configuration. At least one factory configuration has to be created before clicking the run button. Not doing so, will result in failure of the task.



iii. Go to **CRXDE Lite** and create a node of type **sling:OsgiConfig** within an appropriate folder of apps. You can use the existing configuration folder **/apps/geometrixx/config**, or you can create your own config folder in another path: **/apps/training/config**.

Node:

Node Type = sling:OsgiConfig

Name = com.adobe.granite.workflow.purge.Scheduler-WEEKLY

Properties:

scheduledpurge.name	String = Weekly Purge
scheduledpurge.workflowStatus	String = COMPLETED
scheduledpurge.modelIds	String[] = ""(leave this blank)
scheduledpurge.daysold	Long = 28

After you create the node, you should find the new configuration in the Web Console under the heading **Adobe Granite Workflow Purge Configuration**.

iv. Now, run the **Workflow Purge** task in the **Weekly Maintenance Window** and you'll see the card turn green with the note, 'succeeded on ...'

NOTE: For testing purposes, you can quickly fill in the form in the Web Console configuration tab, but it is preferable to use a node of type sling:OsgiConfig with the appropriate name and properties as described on the left.

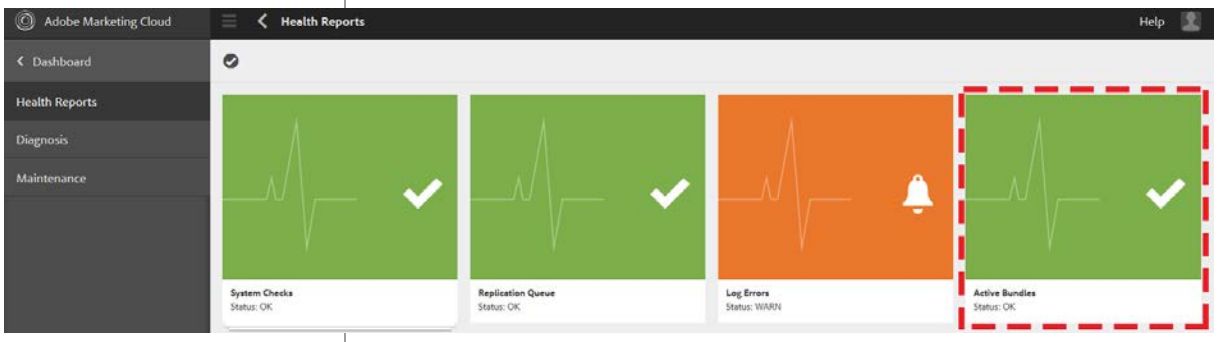
After carrying out the three tasks described above, you can return to **Health Reports** and you will see that the **System Checks** tab has changed to green with a status of **OK**. You can also see three tasks with the green 'OK' status on the 'System Checks' page:

Congratulations! You successfully used the Operations dashboard to check the status of your AEM instance and diagnose issues, add/modify configurations, and run tasks.

Exercise 19.2
Work With Active Bundles

In this exercise, you go through an example of how to use a health report to check the status of active bundles and to change the configuration to include only the bundles you want to check.

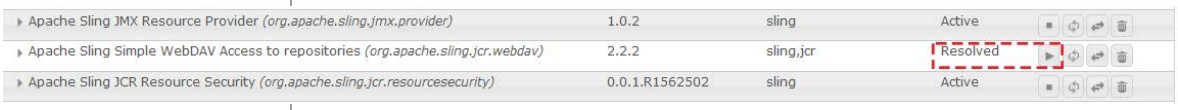
1. Check the **Active Bundles** card in the **Health Reports** tool. After the vanilla installation of AEM with the out-of-the-box settings, the status of this card should be **OK** (Green card).



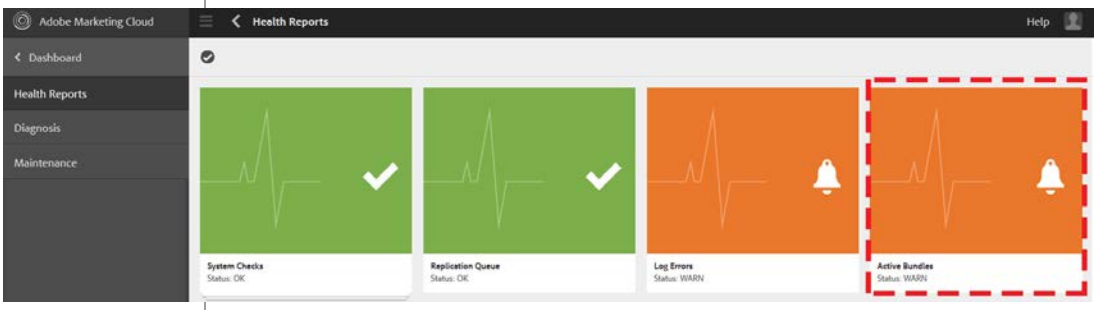
Click the **Info** tab to check the current status and note that the status is **OK** and that there are 'No inactive or unresolved bundles.'

2. The next step is to deactivate a bundle and see how this affects the **Active Bundles** health.
   a. Go to the **Web Console** bundles tab (http://localhost:4502/system/console/bundles) and stop the bundle titled **Apache Sling Simple WebDAV Access to repositories**.
   Click the **Stop** button to stop the bundle:

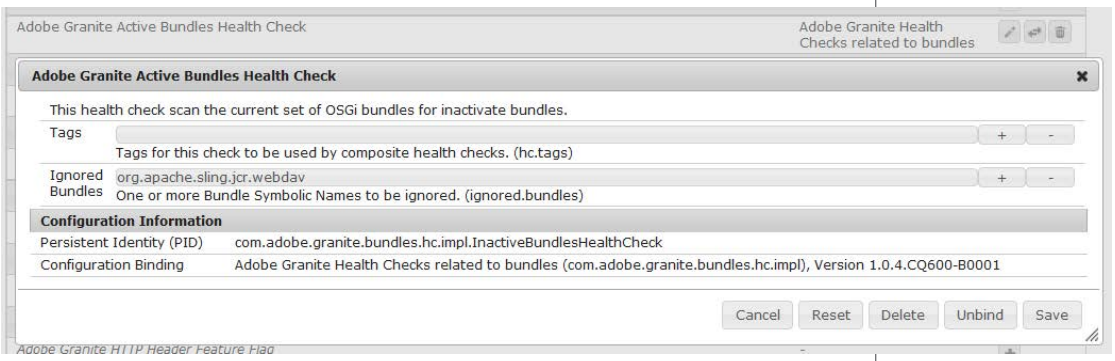
This will change the status from 'active' to 'resolved':



   b. Return to the **Health Reports** tab in the AEM Dashboard and you will notice that the **Active Bundles** card has changed to the orange 'WARN' status:



3. You can now modify the list of bundles that are checked by the **Active Bundles** health check.
   a. To find the appropriate configuration for this health check, go to the Web Console and search for the configuration panel, **Adobe Granite Active Bundles Health Check**. There is also a short cut for this: Click the configuration button on the **Active Bundles** card that appears in step 1.



For testing purposes, you can quickly fill in the form in the Web Console configuration tab, but it is preferable to use a node of type **sling:OsgiConfig** with the appropriate name and properties as described below.

i. Go to CRXDE Lite and create a node of type **sling:OsgiConfig** within an appropriate folder of apps. You can use the existing configuration folder **/apps/geometrixx/config**, or you can create your own config folder in another path: **/apps/training/config**.

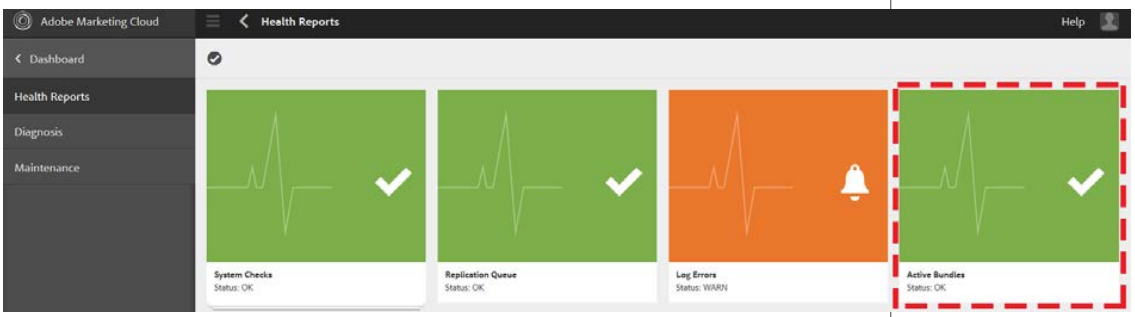
Node:

Node Type	sling:OsgiConfig
Name	com.adobe.granite.bundles.hc.impl.inactiveBundlesHealthCheck

Properties:

hc.tags *	String[]= "" (leave this blank)
ignored.bundles	String[]= org.apache.sling.jcr.webdav

ii. Return to the **Health Reports** tab in the AEM dashboard and you will notice that the **Active Bundles** card has returned to the green **OK** status:



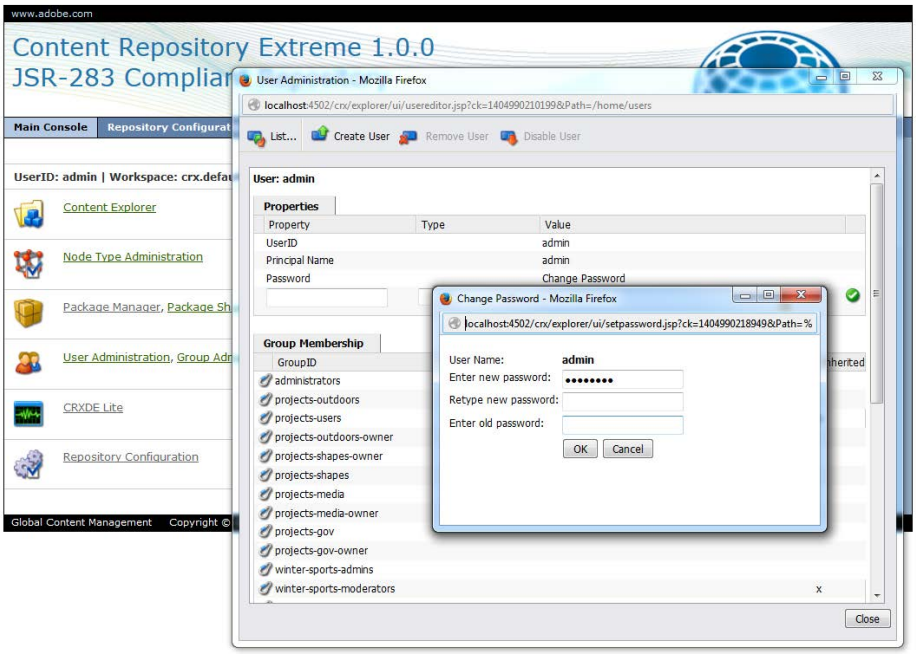
Congratulations! You successfully used the Operations dashboard to check the status of your AEM instance concerning the 'Active Bundles', and you have learned how to configure the health check bundle associated with this card.

Exercise 19.3
Work With Security Checks

1. Consult the **Security Checks** dashboard, and you will see some cards for a number of key security points. Note the disclaimer as it is important to realize that there is more to security than what you find in the dashboard. For more help on this point, you should consult the online documentation as concerns the Security checklist:
   http://docs.adobe.com/docs/en/aem/6-0/administer/security/security-check-list.html
   Most of the cards in this dashboard are in the orange 'WARN' status. We will show how to interpret and modify the points concerning one of the cards, the **Default Login Accounts** card.
   a. Go to the **Security Checks** dashboard and notice that the **Default Login Accounts** card has the orange 'WARN' status. By clicking the **Info** button in the card, you can see information about addressing these issues:

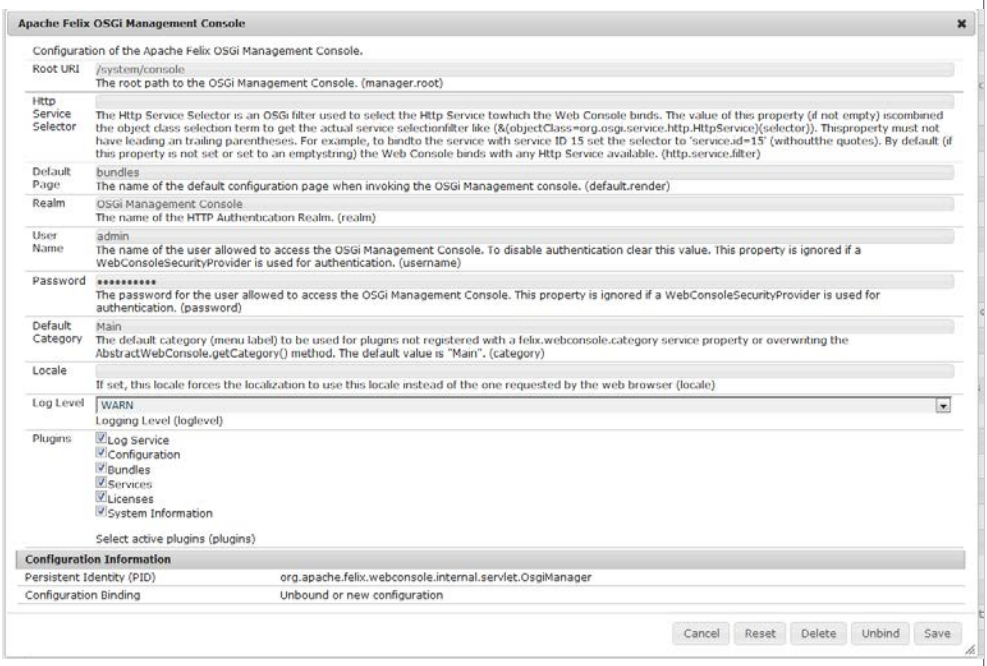
Health Check	Status: WARN
[WARN]	Login as [admin] succeeded, was expecting it to fail.
[WARN]	Login as [author] succeeded, was expecting it to fail.
[WARN]	Checked 2 account logins, 2 matched. Failure.
[DEBUG]	It is strongly recommended to change the default admin accounts for CQ.

   Let us address the issues by following the hints.
   i. Change the CRX admin password:
      Go to http://localhost:4502/crx/explorer in **User Admin** and change the password for the admin user.



ii. Change the OSGi admin password:

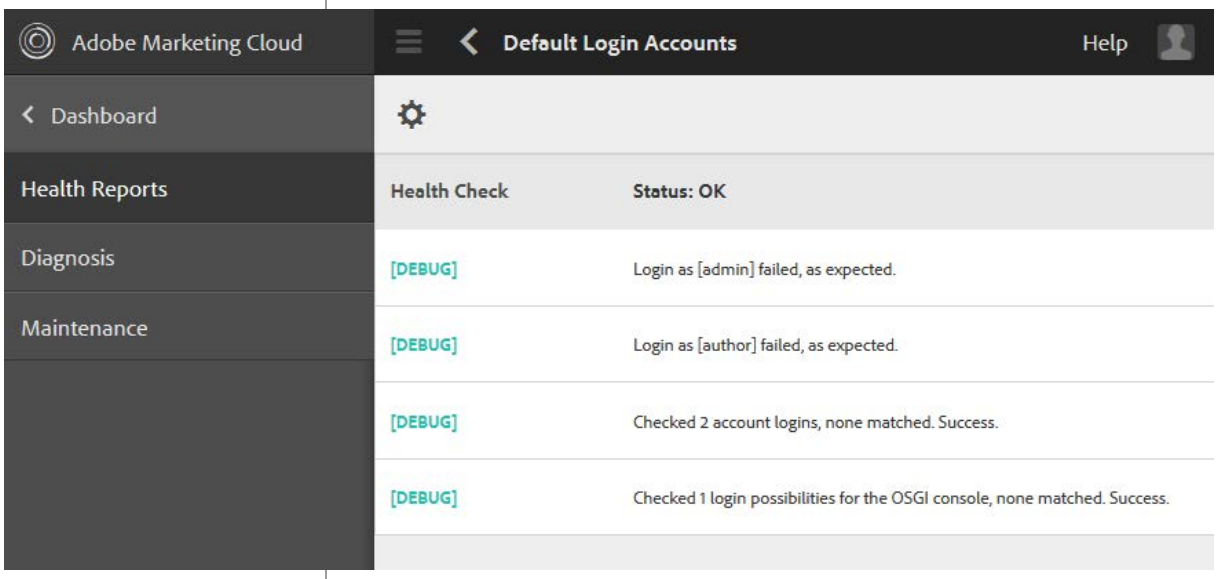
Go to the Web Console configurations page and change the admin password for the configuration titled **Apache Felix OSGi Management Console** directly in the console.



NOTE: This is one configuration that must not be done with a sling:OsgiConfig node.

iii. Change the CRX user **Author** password or delete this user:
You can use the Touch UI for this change:
http://localhost:4502/libs/granite/security/content/userEditor.html/home/users/geometrixx/author
Or you can use the Classic UI:
http://localhost:4502/useradmin

c. You may now return to the **Security Checks** dashboard and notice that the **Default Login Accounts** card has changed to the green **OK** status. By clicking the **Info** button in the card, you can see that the default logins have been changed:



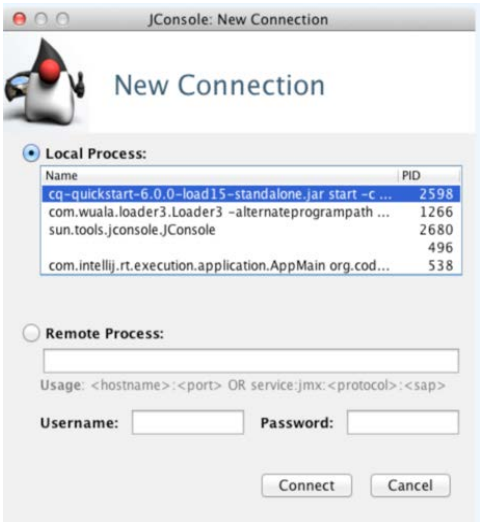
Congratulations! You consulted the Security Checks dashboard, addressed some of the points in the Security checklist to change the status of the Default Login's security card, and learned about the Security checklists.

Monitoring

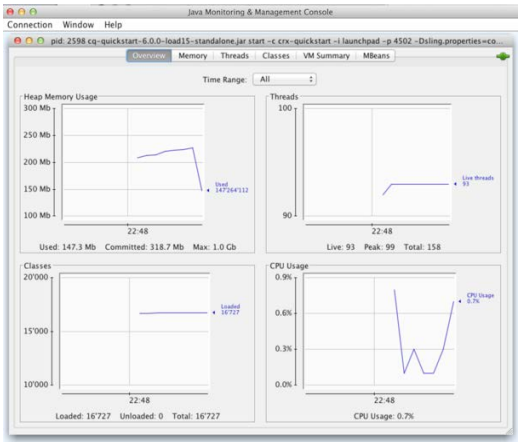
This only works with Java 7 when followin is sent to:
JVM -Dcom.sun.management.jmxremote-Dcom.sun.management.jmxremote.local.only=false.

- For monitoring purposes, AEM 6.1 exposes JMX Managed Beans (MBeans) much like the previous versions of AEM did.
- The Jmx data can be accessed by specialized monitoring software.
- To manually inspect the JMX data, use the jconsole tool, which is provided with your Java distribution.
- Start jconsole on the server where AEM 6.1 is running.

The AEM process should be visible:



After connecting to the AEM process, you should see the monitoring data exposed:



External Login Modules

Oak has a new concept of External Login Modules. You can set up the same within Oak as explained in following sections.

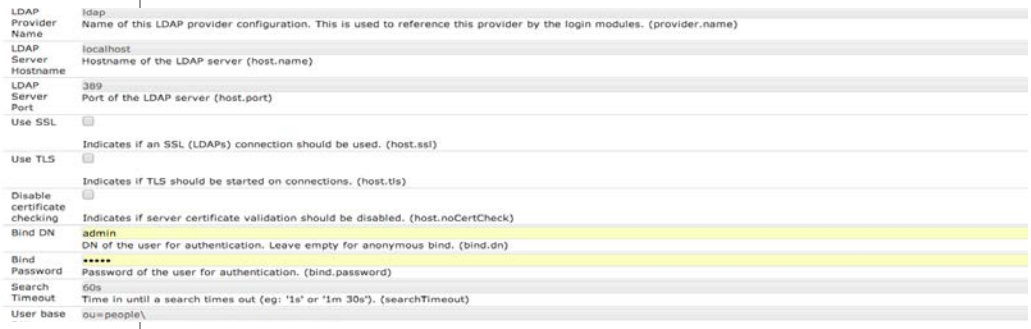
JAAS login modules can be deployed as OSGi bundles with startup level 15.

The external login module can be activated by configuring it in the AEM system console at http://localhost:4502/system/console/configMgr, in the Apache Jackrabbit Oak External Login Module section.

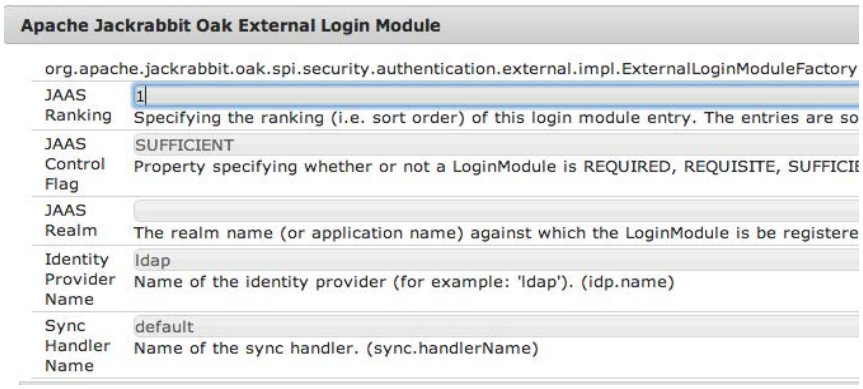
Configuration of the synchronization of the user data with the external login module is configured in the Apache Jackrabbit Oak Default Sync Handler section.

LDAP Configuration

LDAP login is implemented directly in Oak as a module.
With this, the configuration of LDAP has significantly changed; the configuration can be made now in the system console at http://localhost:4502/system/console/configMgr, in the Apache Jackrabbit Oak LDAP Identity Provider section.
The configuration screen contains standard LDAP settings:



In order to activate the LDAP module, the LDAP login module has to be entered in the Apache Jackrabbit Oak External Login Module settings by setting the Identity Provider Name configuration to LDAP:



Use JMX and the MBeans Provided by CRX

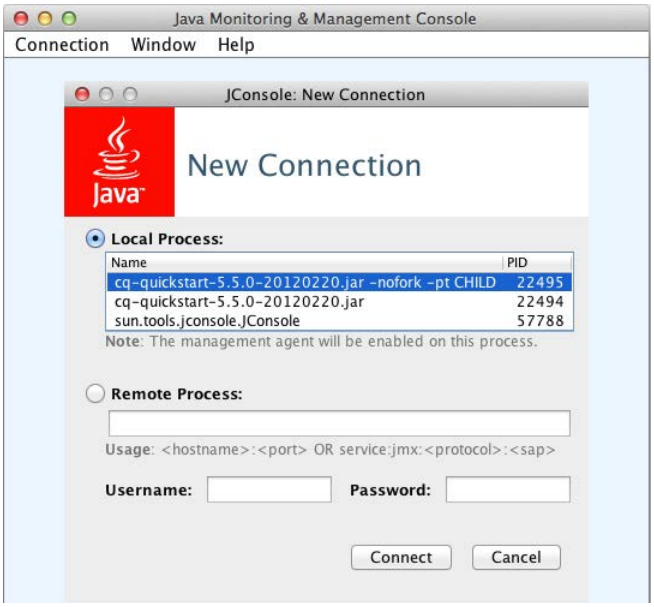
CRX allows external applications to interact with MBeans through JMX. Using generic consoles, such as JConsole or domain-specific monitoring applications, allows getting and setting CRX configurations and properties, and the monitoring of performance and resource usage.

Use JConsole to connect to CRX

In order to connect to CRX using JConsole, follow these steps:

1. Open a Terminal window.
2. Enter the following command:
   `jconsole`

JConsole starts and the JConsole window appears. JConsole displays a list of local JVM processes. The list will contain two quickstart processes. Select the quickstart 'CHILD' process from the list of local processes (usually the one with the higher PID).



Connect to a Remote CRX Process

In order to connect to a remote CRX process, the JVM that hosts the remote CRX process will need to be enabled to accept remote JMX connections.

To enable remote JMX connections, the following system property must be set when starting the JVM:

```
com.sun.management.jmxremote.port=portNum
```

In the property above, portNum is the port number through which you want to enable JMX RMI connections. Be sure to specify an unused port number. In addition to publishing an RMI connector for local access, setting this property publishes an additional RMI connector in a private read-only registry at the specified port using a well-known name, 'jmxrmi'.

By default, when you enable the JMX agent for remote monitoring, it uses password authentication based on a password file that needs to be specified using the following system property when starting the Java VM:

```
com.sun.management.jmxremote.password.file=pwFilePath
```

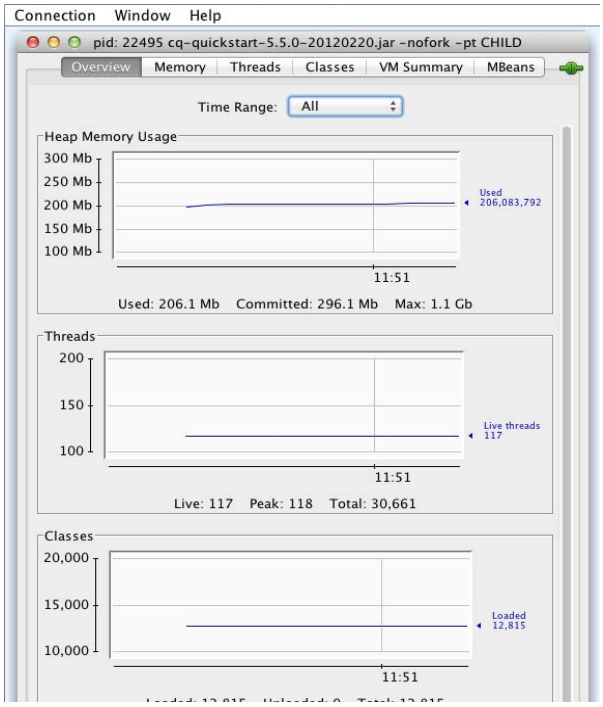
See the relevant JMX documentation for detailed instructions on setting up a password file. You can access more details at http://docs.oracle.com/javase/6/docs/technotes/guides/management/agent.html.

Example:

```
$ java
  -Dcom.sun.management.jmxremote.password.file=pwFilePath
  -Dcom.sun.management.jmxremote.port=8463
  -jar ./cq-quickstart.jar
```

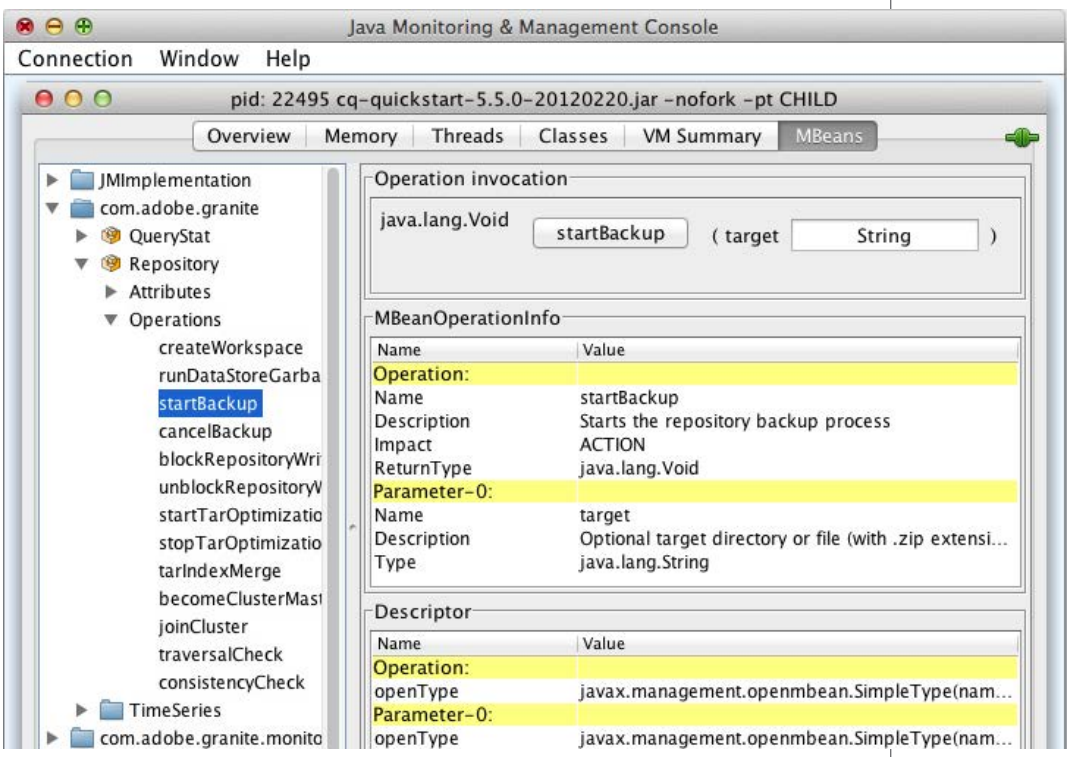
Use the MBeans Provided by CRX

After connecting to the quickstart process, JConsole provides a range of general monitoring tools for the JVM that CRX is running in.



In order to access CRX's internal monitoring and configuration options, go to the MBeans tab, and from the hierarchical content tree on the left, select the Attributes or Operations section that you are interested in; for example, the com.adobe.granite/Repository/Operations section.

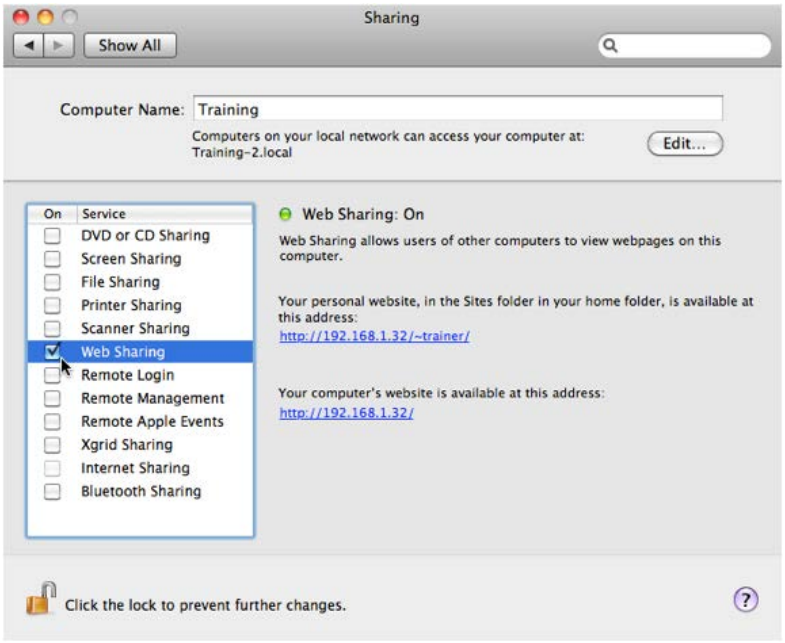
Within that section, select the desired attribute or operation in the left pane.



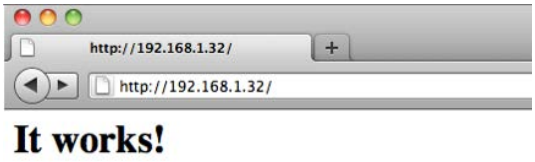
Configuring the AEM Dispatcher on Mac OS X

The following instructions describe how to configure the AEM Dispatcher on a Mac OS X version 10.[6-7].X, and the Apache Web Server 2.2 that comes already installed on Mac OS X.

Ensure that the Apache Web Server already installed in your Mac OS X is up and running on your system. Enter your system preferences, click on sharing, and then start your 'Web Sharing' by selecting its check box.



You should be able to browse on your computer as follows:

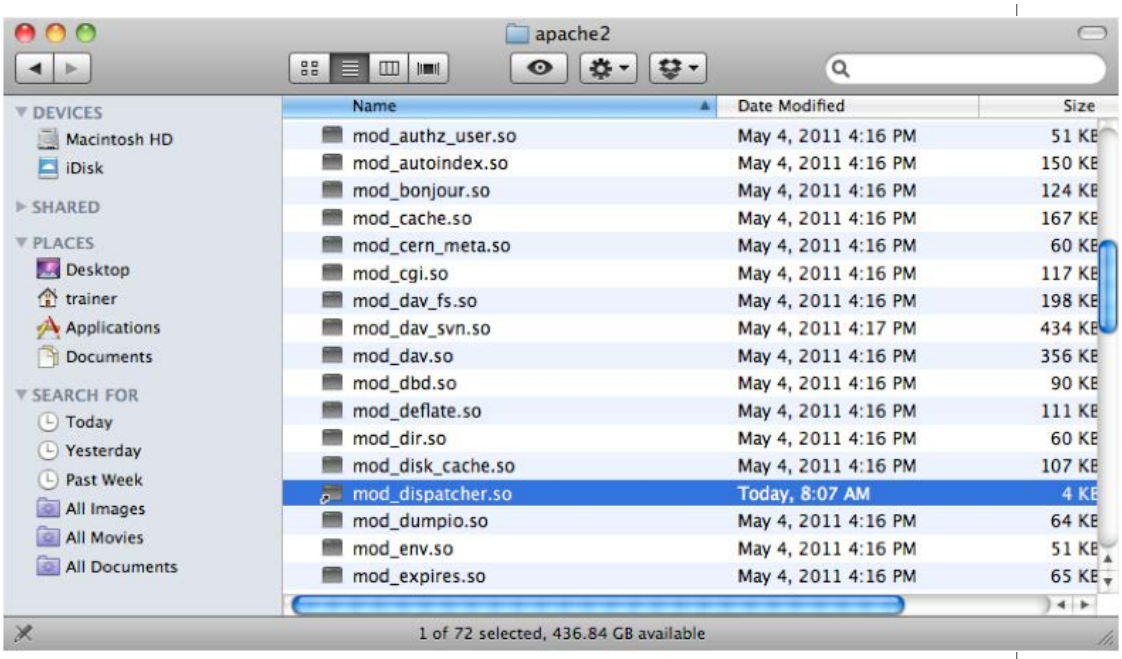


Install the Dispatcher Module In Apache

1. Open the Terminal (command window). The Terminal application is located under **/Applications/Utilities/Terminal.app**.
2. Enter the following command to verify the installed Apache Web Server version.
   $ httpd -v
3. Unzip the latest CQ Dispatcher build, appropriate for your operating system, to a temporary directory. The CQ Dispatcher files are located on the memory stick under **<USB>/distribution/dispatcher** (dispatcher-apache2.2-darwin-x86-64-4.1.0.tgz). If the appropriate Dispatcher is not in the memory stick, you may download it from the package share: **http://localhost:4502/crx/packageshare/**.
4. Open the Terminal.
5. Enter the following commands to open the normally hidden folders in the Mac OS X Finder, i.e. /usr and /private folders:
   $ open /usr
   $ open /etc
6. In the new Finder window of **/usr**, navigate to the **/usr/libexec/apache2** folder. Copy the **dispatcher-apache2.2-4.1.0.so** file from the unpacked archive to this location.
7. Then, you should create a symbolic link in order to have an easier name to be useed in your scripts. In the Terminal window, type the following:
   cd /usr/libexec/apache2/ and then create the 'symbolic link':
   ln -s dispatcher-apache2.2-4.1.0.so mod_dispatcher.so

NOTE: You may not have the rights to do so; in that case, you have to use the sudo command (to launch commands as root on your machine).
sudo ln -s dispatcher-apache2.2-4.1.0.so mod_dispatcher.so

After doing so, you will be able to see the mod_dispatcher.so file in the /usr/libexec/apache2/ folder in the Finder.



Finder Showing the CQ Dispatcher Module

8. Next, in the Finder window of **/etc**, navigate to **/etc/apache2** and copy the **dispatcher.any** file from the unpacked Dispatcher archive to this location.

Configure httpd.conf

Configure Apache for the Dispatcher. In the /private/etc/apache2 folder, you will find the httpd.conf file (we are using the default Apache server that comes with Mac OS X). You can also use the httpd.conf file that comes with the Dispatcher archive from the USB memory stick.

1. Add the following text to the **httpd.conf** file at the end of the first Load Module section (somewhere before the #Apple specific modules):

```
# LoadModule foo_module libexec/mod_foo.so
# Add to the end of the LoadModule section
LoadModule dispatcher_module libexec/apache2/mod_dispatcher.so
```

2. Add the following text to the **httpd.conf** file at the end of the Load Module section, right after the ServerAdmin you@example.com setting.

```
<IfModule disp_apache2.c>
   # location of the configuration file. eg: 'conf/dispatcher.
   any'
       DispatcherConfig  /etc/apache2/dispatcher.any
    # location of the dispatcher log file. Just an example,
   use var'
       DispatcherLog     /etc/apache2/dispatcher.log
       # log level for the dispatcher log
```

```
        # 0 Errors
        # 1 Warnings
        # 2 Infos
        # 3 Debug
        DispatcherLogLevel 3.
        DispatcherNoServerHeader 1
        DispatcherDeclineRoot 0
      </IfModule>
```

3. Also, add the **SetHandler** statement to the context of your configuration (<Directory>, <Location>) for the Dispatcher to handle the incoming requests. The simplest setting is to replace the entire default directory configuration.
   First, we configure the 'default' to be a very restrictive set of features.

```
<Directory />

        <IfModule disp_apache2.c>

                SetHandler dispatcher-handler
            ModMimeUsePathInfo On

        </IfModule>

             Options FollowSymLinks
             AllowOverride None
</Directory>
```

Configure the Dispatcher

Follow the instructions for the 'Configure the Dispatcher' section with the following exceptions:

When configuring the cache, use the following /docroot:

```
/cache
       {
       # The docroot must be equal to the document root of
    the webserver. The
       # dispatcher will store files relative to this
    directory and subsequent
       # requests may be “declined” by the dispatcher,
    allowing the webserver
       # to deliver them just like static files.
       /docroot “/Library/WebServer/Documents”
```

Update the Publish Instance Port Number on the dispatcher.any File

```
/renders
       {
       /rend01
         {
         # Hostname or IP of the render
         /hostname “127.0.0.1”
         # Port of the render
         /port “4503”
         # Connect timeout in milliseconds, 0 to wait
    indefinitely
         # /timeout “0”
         }
        }
```

The http server process must have read/write access to that folder in order to write the cache files. You can choose another folder, but you then have to be sure that the httpd server daemon has read/write access to /Library/WebServer/Documents (use chown and chgrp if necessary).

4. Restart Apache by issuing the following command from a Terminal window: **$sudo apachectl graceful Restart “Web Sharing”**
5. Access the Geometrixx website using the following URLs:

- Author instance**:** http://localhost:4502/content/geometrixx.html
- Publish Instance**:** http://localhost:4503/content/geometrixx.html
- Web Server/Dispatcher**:** http://localhost/content/geometrixx.html

Test your configuration: The cached pages should be showing up in the web server's document root. If you now navigate to the web server document root directory, you should see directories and files appear as the CQ Dispatcher and web server begins to cache pages.

Congratulations! You successfully configured the CQ Dispatcher with the Apache Web Server already installed on your Mac OS X, cached pages in the web server document root, and accessed activated content pages from the web server.

Dispatcher Installation on a Different Version of Microsoft IIS

wFor information on how to install this web server, see the following resources:

- Microsoft's own documentation on IIS
- The official Microsoft IIS site

Required IIS Components

IIS versions 7.5, 8.0, and 8.5 require that the following IIS component is installed:

- ISAPI extensions

Also, you must add the Web Server (IIS) role. Use Server Manager to add the role and components.

Microsoft IIS - Installing the Dispatcher Module

The required archive for Microsoft IIS is:

- dispatcher-iis-<operating-system>-<*dispatcher-release-number*>.zip

The zip file contains the following files:

File	Description
disp_iis.dll	The Dispatcher dynamic link library file
disp_iis.ini	Configuration file for IIS; this example can be updated with your requirements NOTE: The ini file must have the same name-root as the dll.
dispatcher.any	An example configuration file for the Dispatcher
author_dispatcher.any	An example configuration file for Dispatcher working with the author instance
release-notes.txt	Installation instructions, a list of fixed issues in the current and previous releases, and other last-minute information NOTE: Read this file before installing.

Use the following procedure to copy the Dispatcher files to the correct location.

1. Use Windows Explorer to create the **<IIS_INSTALLDIR>/Scripts** directory, for example, **C:\inetpub\Scripts**.

2. Extract the following files from the Dispatcher package into this **Scripts** directory:
   - **disp_iis.dll**
   - **disp_iis.ini**

One of the following files depending on if Dispatcher is working with an AEM Author instance or Publish Instance:
Author instance: author_dispatcher.any
Publish Instance: dispatcher.any

Microsoft IIS - Configure the Dispatcher INI File

Edit the disp_iis.ini file to configure the Dispatcher installation. The basic format of the .ini file is as follows:

```
[main]
configpath=<path to dispatcher.any>
loglevel=1|2|3
servervariables=0|1
replaceauthorization=0|1
```

The following table describes each property.

Parameter	Description
configpath	The location of dispatcher.any within the local file system (absolute path)
logfile	The location of the dispatcher.log file; if this is not set, then log messages go to the Windows event log
loglevel	Defines the log level used to output messages to the event log; the following values may be specified: 0: Error messages only. 1: Errors and warnings. 2: Errors, warnings, and informational messages. 3: Errors, warnings, informational messages, and debug messages. Note: It is recommended to set the log level to 3 during installation and testing, and then revert to 0 when running in a production environment.
replaceauthorization	Specifies how authorization headers in the HTTP request are handled; the following values are valid: 0: Authorization headers not modified. 1: Replaces any header named 'Authorization' other than 'Basic' with its 'Basic <IIS:LOGON_USER>' equivalent.

servervariables	Defines how server variables are processed: 0: IIS server variables are sent to neither the Dispatcher nor AEM. 1: All IIS server variables (such as LOGON_USER, QUERY_STRING, ...) are sent to the Dispatcher, together with the request headers (and also to the AEM instance if not cached). Server variables include AUTH_USER, LOGON_USER, HTTPS_KEYSIZE, and many others. See the IIS documentation for the full list of variables, with details.

An example configuration:

```
[main]
configpath=C:\Inetpub\Scripts\dispatcher.any
loglevel=1
servervariables=1
replaceauthorization=0
```

Configure Microsoft IIS

Configure IIS to integrate the Dispatcher ISAPI module. In IIS, you use wildcard application mapping.

Configure Anonymous Access - IIS 7.5, 8.0, and 8.5

The default Flush Replication Agent on the Author instance is configured so that it does not send security credentials with flush requests. Therefore, the website that you are updating using a Dispatcher cache must allow anonymous access.
If your website uses an authentication method, the Flush Replication Agent must be configured accordingly.

1. Open IIS Manager and select the website that you are using as the Dispatcher cache.
2. Using Features View mode, in the **IIS** section, double-click **Authentication**.
3. If Anonymous Authentication is not enabled, select **Anonymous Authentication**, and in the **Actions** area, click **Enable**.

Integrate the Dispatcher ISAPI Module - IIS 7.X, 8.X

Use the following procedure to add the Dispatcher ISAPI Module to IIS.

1. Open IIS Manager.
2. Select the website that you are using as the Dispatcher Cache.
3. Using Features View mode, in the **IIS** section, double-click **Handler Mappings**.
4. In the **Actions** panel of the **Handler Mappings** page, click **Add Wildcard Script Map**, add the following property values, and then click **OK**:
   - Request Path: *****
   - Executable: The absolute path of the **disp_iis.dll** file, for example **C:\inetpub\Scripts\disp_iis.dll**.

- Name: A descriptive name for the handler mapping, for example, **Dispatcher**.

5. In the dialog box that appears, to add the **disp_iis.dll** library to the **ISAPI and CGI Restrictions** list, click **Yes**.
   For IIS 7.0 and 7.5, the configuration is complete. Continue with the remaining steps if you are configuring IIS 8.0.
6. (IIS 8.0) In the list of handler mappings, select the one that you just created, and in the **Actions** area, click **Edit**.
7. (IIS 8.0) In the **Edit Script Map** dialog box, click the **Request Restrictions** button.
8. (IIS 8.0) To ensure that the handler is used for files and folders that are not yet cached, clear **Invoke Handler Only If Request Is Mapped To**, and then click **OK**.
9. (IIS 8.0) On the **Edit Script Map** dialog box, click **OK**.

Configure Access to the Cache - IIS 7.5, 8.0, and 8.5

Provide the default App Pool user with write access to the folder that is being used as the Dispatcher cache.

1. Right-click the **root** folder of the website that you are using as the Dispatcher cache and click **Properties**, such as **C:\inetpub\wwwroot**.
2. On the **Security** tab, click **Edit**, and then in the **Permissions** dialog box, click **Add**. A dialog box opens for selecting user accounts. Click the **Locations** button, select your computer name, and then click **OK**.
   Keep this dialog box open while you complete the next step.
3. In IIS Manager, select the IIS site that you are using for the Dispatcher cache, and on the right side of the window, click **Advanced Settings**.
4. Select the value of the **Application Pool** property and copy it to the clipboard.
5. Return to the open dialog box. In the **Enter The Object Names To Select** box, type **IIS AppPool**, and then paste the contents of your clipboard.
   The value should look like the following example:
   IIS AppPool\DefaultAppPool
6. Click the **Check Names** button. When Windows resolves the user account, click **OK**.
7. In the **Permissions** dialog box for the **dispatcher** folder, select the account that you just added, enable all of the permissions for the account **except for Full Control**, and click **OK**. Click **OK** to close the folder **Properties** dialog box.

Register the JSON Mime Type - IIS 7.5, 8.0, and 8.5

Use the following procedure to register the JSON MIME type, when you want Dispatcher to allow JSON calls.

1. In IIS Manager, select your website and using Features View, double-click **Mime Types**.
2. If the JSON extension is not in the list, in the **Actions** panel, click **Add**, enter the following property values, and then click **OK**:
   - File Name Extension: **.json**
   - MIME Type: **application/json**

Remove the bin Hidden Segment - IIS 7.5, 8.0, and 8.5

Use the following procedure to remove the bin hidden segment. Websites that are not new can contain this hidden segment.

1. In IIS Manager, select your website and using Features View, double-click **Request Filtering**.
2. Select the **bin** segment, click **Remove**, and in the confirmation dialog box, click **Yes**.

Log IIS Messages to a File - IIS 7.5, 8.0, and 8.5

Use the following procedure to write Dispatcher log messages to a log file instead of to the Windows event log. You need to configure Dispatcher to use the log file, and provide IIS with write access to the file.

1. Use Windows Explorer to create a folder named **dispatcher** below the **logs** folder of the IIS installation. The path of this folder for a typical installation is **C:\inetpub\logs\dispatcher**.
2. Right-click the **dispatcher** folder and click **Properties**.
3. On the **Security** tab, click **Edit**, and then in the Permissions dialog box, click **Add**. A dialog box opens for selecting user accounts. Click the **Locations** button, select your computer name, and then click **OK**.
   Keep this dialog box open while you complete the next step.
4. In IIS Manager, select the IIS site that you are using for the Dispatcher cache, and on the right side of the window, click **Advanced Settings**.
5. Select the value of the **Application Pool** property and copy it to the clipboard.
6. Return to the open dialog box. In the **Enter The Object Names To Select** box, type **IIS AppPool** and then paste the contents of your clipboard. The value should look like the following example:
   IIS AppPool\DefaultAppPool
7. Click the **Check Names** button. When Windows resolves the user account, click **OK**.

8. In the **Permissions** dialog box for the **dispatcher** folder, select the account that you just added, enable all of the permissions for the account **except for Full Control**, and click **OK**. Click **OK** to close the folder **Properties** dialog box.
9. Use a text editor to open the **disp_iis.ini** file.
10. Add a line of text similar to the following example to configure the location of the log file and then save the file:

```
logfile=C:\inetpub\logs\dispatcher\dispatcher.log
```

Running a MongoDB Instance, Clustering in AEM Upgrade/ Migration to AEM 6.0

Start a Local MongoDB Instance

On Mac OS/Unix, execute:

```
ulimit -n 2048
```

On Windows, change the directory to the instance folder and start MongoDB with the following command:

```
mongod --config crx-quickstart/conf/mongo.conf
```

Start a AEM 6.0 Instance with MongoMK

Open a new shell, change the current directory to the instance folder, and start the AEM installation with the following command:

```
java -jar aem-6.0.0.20140515-generic.jar -r
    author,crx3,crx3mongo -p 4502
```

Afterward, check that there is no 'crx-quickstart/repository' folder (this would mean that you have installed TarMK); wait for the AEM installation to complete.
Wait for browser startup. Meanwhile, you can check the log files 'mongo.log' and 'error.log'.

Verify Installation

To check for a successful MongoDB installation, execute the following steps:

1. Start a Mongo shell with the following command:
   `mongo <host>:<port>`
2. In the shell, execute the command:
   `db.stats()`

This should show a non-empty database 'OakAuthor'.

3. To check for a successful AEM 6.0 installation, open CRXDElite, log in as **admin**, and see if the folder structure is present. Create a **sling:folder**, set a property, and save those changes.

Cluster Setup

First, stop all instances of AEM 6.0 and MongoDB.

Then, create at least two more MongoDB instances. For this, you need a new mongo.conf for each new MongoDB instance. Each config should point to a different data directory ('dbpath'), where the instance stores its data and contains the line.

These instances can be on the same or on different machines. If they are on different machines, check if the new instances are reachable from the first instance, and check that all instances can resolve the host name configured in the value of 'bind_ip' of all other instances.

Finally, start the new instances.

When all MongoDB instances are running, connect the mongo shell to the initial MongoDB instance, which already contains the AEM 6.0 repository, and execute the following commands:

```
mongo localhost:27017/OakAuthor
```

NOTE: Only execute the following commands on the initial MongoDB.

To initialize the Replica Set, execute the following command:

```
rs.initiate()
```

See the Replica Set config, it should show only one member:

```
rs.conf()
```

Add all other MongoDB nodes to the Replica Set (NOTE: the quotes are literal):

```
rs.add("<other_host>:<port>")
```

Check Replica Set configuration:

```
rs.conf()
rs.status()
```

Validate the cluster config. The commands above should show the initial instance plus the instances you just added. The initial instance should be in the PRIMARY state; all other instances should be in the STARTUP2 or SECONDARY state. See MongoDB documentation for a description of these states.

Configure AEM 6.0

Finally, configure AEM 6.0 to use the MongoDB cluster.

Start all instances you want to use. Currently, they will not run because there is no repository configured. To fix this, open the Felix console at:

```
http://<aem_host>:<port>/system/console/configMgr
```

Then configure the **org.apache.jackrabbit.oak.plugins.document.DocumentNodeStoreService**.

Change the **mongouri** property to:

```
mongodb://<host1>:<port>,<host2>:<port>,<hostN>:<port>
```

This is where you list all the MongoDB instances in your Replica Set. See rs.status() for a list of all instances.

Congratulations! You just set up your first MongoDB cluster with AEM 6.0.

Upgrade to AEM 6.0

The upgrade plan covers various paths from which an upgrade to AEM 6.0 can be achieved.

Because AEM 6.0 supports both CRX3 and CRX2 run mode, the upgrade path from the previous versions (CQ 5.2.1, 5.3, 5.4, or CQ 5.6.1 to AEM 6.0) is primarily decided considering the run mode and database.

There are two distinct steps for a full upgrade:

1. **In-Place Upgrade:** Drop in the new jar file in place of the original one in your <aem-install> directory, (right next to your existing crx-quickstart directory) and launch it. This upgrades the system, but leaves the existing CRX2 repository as is.
2. **Migrating the Repository to Oak:** When the in-place upgrade is complete, you can optionally migrate the existing CRX2 repository to the new Jackrabbit Oak-based CRX3 repository.

Upgrade and Use CRX3 Run Mode

1. TarMK as a repository:
   - Upgrade from 5.6.1 or a previous version to 6.0 in CRX2 run mode.
   - Use the migration tool to switch the repository at the backend.
2. MongoDB as a repository:
   - Upgrade from 5.6.1 or a previous version to 6.0 to TarMK as a repository.
   - Migrate to Mongo DB.

	TarPM [CQ 5.6.1 to AEM 6.0]	TarMK [CQ 5.6.1 to AEM 6.0]	MongoDB [CQ 5.6.1 to AEM 6.0]
OS: Windows 2008 and RHEL 6	Run AEM with CRX2.0, that is, do not migrate to CRX3	Run AEM in CRX3 mode	Run AEM in CRX3 mode
Types	In-place upgrade	Migration utility after in-place upgrade	Migrate to MongoDB
Mode	Standalone	Standalone	Standalone

Plan Your Upgrade

This section provides information and guidelines for planning and executing your upgrade to AEM 6.0, from initial preparation and tests, through to implementation on your production environments.

Areas of Risk

When preparing the upgrade, you need a list of all points where the project extended or customized the product. Such points include custom components, overlays and extensions of foundation components, special selectors, overlaid widgets, and many

others. This list should be available from the original project development. If the project did not explicitly list all extension/customization points, this is a good time to start such a list for the future.

Preparation and Testing

It is recommended to start by making a copy of your production Author instance. Use this to re-create the environment on a separate server (preferably similar hardware). You can then run the upgrade procedure on this environment to:

- identify risks specific to your instance
- test the upgrade procedure on your project
- make initial checks to confirm integrity of content and functionality on the resulting AEM 6.0 instance
- refine the time estimations for your individual project and hardware

You can then use this upgraded instance to uncover any changes needed to your customized code. When these changes are complete, you can export the upgraded code base to SVN for use by your QA team to verify the functionality.

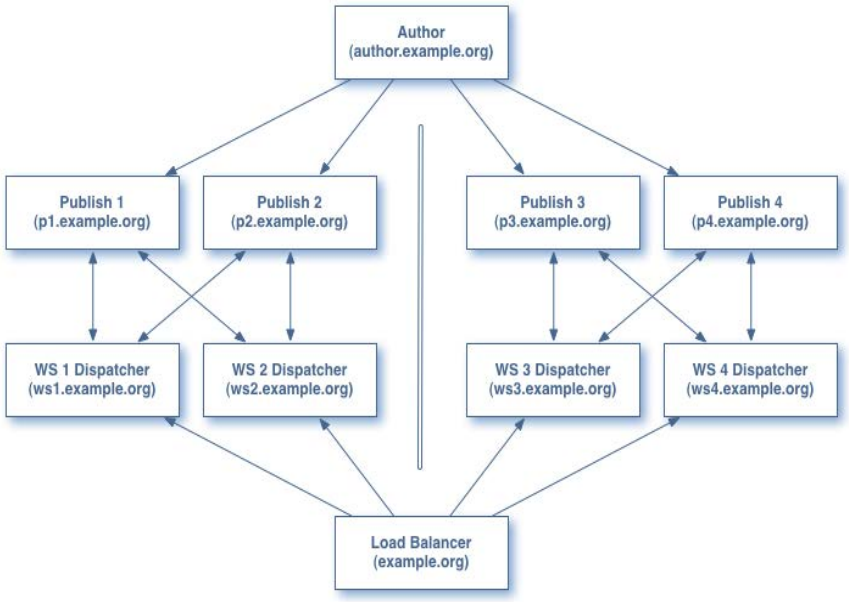
NOTE: For customized projects, the process of upgrading and testing may need to be repeated until results and individual steps are confirmed.

Ongoing development might require a separate branch for performing the code upgrade. If necessary, make sure to define synchronization points for the developer upgrade, QA tests, and merging when complete.

Upgrade Your Production Environments

After fully testing the upgrade for your instance, you need to plan how this will be carried out on your production environments—both author and publish.

The following diagram shows an overview of a sample system architecture that will be used as the basis for a sample upgrade plan:



This sample system is distributed across two geographically separate data centers. The distance between them is significant enough to introduce network latency. The sequence of upgrades is:

- **Upgrade the author instance:**
  The author instance will be upgraded and tested before the publish instances are upgraded. This allows you to focus on one upgrade. When that is finished, authors can start using it again while the upgrade team works on the publish instances (with the restriction that no publishing is allowed during this time).
- **Upgrade one Publish instance in each location:**
  Due to long upgrade times and network latency between the data centers only one publish is upgraded in each location.
- **Upgrade remaining Publish instances:**
  After that, a copy of the upgraded publish is moved to the other server at the respective data center.

The upgrade of the production system needs careful planning to reduce system downtime and the risk of data loss.

Upgrade to AEM 6.0

Overview

An upgrade enables the automated or partially automated conversion of an existing instance of AEM 5 to AEM 6. In-place upgrades are possible; however, after replacing the jar file, the new 6.0 installation will use the CRX 2 (AEM 5) backend as default. Adobe also provides a repository migration tool for existing customers in order to migrate the repository to the new Apache Oak based implementation in AEM 6.0.

Plan the Upgrade

Even though the upgrade process has been designed to be as simple as possible, any upgrade of a production environment is a significant task. For this reason, it is recommended that you invest time in planning your upgrade thoroughly. For guidance on this, see 'Plan Your Upgrade'.

- **In-place upgrade (using the CRX2 backend)**
- **Migrating the repository to Oak**

Since AEM 6.0, the backend has moved to Oak in an effort to enhance scalability and performance of the content repository. For more information about the architectural concepts of Apache Oak, see 'Introduction to Oak' at http://docs.adobe.com/docs/en/aem/6-0/deploy/upgrade/introduction-to-oak.html. The repository migration tool can be found in this location of the quickstart installation folder:
crx-quickstart/opt/helpers/crx2oak
It currently works with these backend configurations:

- **TarMK**
- **MongoMK**

Use the Helper Script to Migrate the Repository

A helper script is also provided in the quickstart package in order to facilitate the use of the migration tool. It can be found in the installation folder at:
crx-quickstart/opt/helpers/crx2oak.sh
Usage for migrating to TarMK:
crx2oak.sh [repository location]
The repository location is the file system path to the CRX2 repository that needs to be converted to Oak. Thus, a correct form to invoke the script would be:
crx2oak.sh /srv/cq/crx-quickstart/repository
Usage for migrating to MongoMK:
crx2oak.sh -m [location of Mongo database]
The location of the Mongo database must be specified in Connection String URI format. Again, a correct way to invoke the script for a MongoMK migration is:
crx2oak.sh -m mongodb://serverhost:27017/aem-author
The script can be run with several optional parameters:

Parameter	Description
-s	Starts a CQ instance after the migration
-j	Specifies a custom crx2oak.jar file to use for the migration
-q	Specifies a custom quickstart.jar file to use when building the initial crx-quickstart folder
-p	The port to use for CQ startup detection; this is useful for instances when the quickstart is configured to start on a different port due to its name, for example, cqauthor-8080.jar
-i	Backs up the previous crx-quickstart folder by renaming it to crx2-quickstart and replaces it with the newly generated instance
-l	Indicates from where to copy a license.properties file into the current working directory in order to automatically start the instance used for migration; if a license.properties file exists in the current folder, this option is not needed
-h	Shows usage information for the script

NOTE: The script will not work for setups with custom DataStore configurations. For such cases, run crx2oak.jar manually, as described earlier.

Troubleshoot Issues After the Upgrade

- **Issues with starting and shutting down AEM after the upgrade:** After upgrading to the CRX3 backend, AEM sometimes fails to start or shut down properly on systems with Java 6 installed. To work around this, add the following parameters to the JVM when starting AEM:
  -XX:+UnlockDiagnosticVMOptions -XX:+UnsyncloadClass
- **Error messages:** The following error messages will appear in the error.log file after the upgrade and can be safely ignored:
  - *ERROR* [OsgiInstallerImpl] org.apache.jackrabbit.vault.fs.impl.io.DocViewSAXImporter Error while applying access control content. javax.jcr.security.AccessControlException: Principal operators does not exist.
  - *ERROR* [OsgiInstallerImpl] com.adobe.granite.installer.factory.packages.impl.PackageTransformer Error while processing uninstall task. com.day.jcr.vault.packaging.PackageException: org.apache.jackrabbit.vault.packaging.PackageException: Unable to uninstall package. No snapshot present.

DataStore Migration

With AEM 6.0, when running the upgrade script, some decisions about the DataStore migration must be made. There are two major options available:

1. Migrate the whole repository including the binaries and migrate them into the new TarMK and tar-based DataStore.
2. Only the repository data is migrated to TarMK; binaries remain in the FileStore as used in previous versions of AEM.

   Possible reasons for selecting option 2:
   - The file-based DataStore is too large to be migrated; migration would take too long.
   - When testing migration, not migrating the binaries speeds up the migration significantly.

Steps to Upgrade

1. **Upgrade to AEM 6.0 with crx2 run mode keeping the same TarPM repository.**
   a. Enable INFO logging for the error and upgrade logs. This can be done by:
      i. Going to AEM Configuration Manager located at http://localhost:4502/system/console/configMgr
      ii. pressing the + button under Apache Sling Logging Logger Configuration
      iii. creating a new factory configuration that will log INFO events for upgrade.log and error.log
   b. Shut down the AEM 5 instance.
   c. Replace the quickstart with the AEM 6.0 jar.
   d. Run the jar and wait for the upgrade to finish.
   e. Check that the upgrade was successful by confirming that:
      The upgrade.log shows the process was finished:

```
*INFO* [FelixStartLevel] com.adobe.cq.upgradesexecutor.
    Activator UPGRADE FINISHED: 6 CodeUpgradeTasks executed
    (out of 6), total time about 32 seconds
```

   f. error.log shows a 'FrameworkEvent STARTED' event:

```
*INFO* [FelixDispatchQueue] org.apache.felix.framework
    FrameworkEvent STARTED
```

   g. Check that the value for System Start Level is 30 in the System Information console. The console can be reached at http://localhost:4502/system/console/vmstat
   h. All the bundles and workflows are in place.

NOTE: If you plan to run AEM 6.0 with a CRX2 backend, you can simply run the jar at this point. However, note that for all upgrades from versions up to and including AEM 5.5, this needs to be explicitly stated through the 'CRX2' run mode option when the AEM 6.0 jar is first started. To do this, run the quickstart with the following command:
java -jar aem-quickstart.jar -r crx2
If the runmode is not specified, the AEM 6.0 instance will start with a new CRX3 repository instead of the existing CRX2 repository.

2. **Upgrade to AEM 6.0 with CRX2 run mode, and then migrate to the TarMK repository**

There are two options for the storage backend used by Oak:

- TarMK: A Microkernel that uses the tar persistence system for storage
- MongoMK: A Microkernel that uses MongoDB for storage

Prerequisite:

a. **Minimum required Java version:** The migration tool only works with Java versions 7 and up.
b. **Upgraded instance:** Before migrating, make sure you have performed an in-place upgrade to AEM 6.0 by following the procedure described previously.
c. Download the repository migration utility from USB or from following link:
https://wiki.day.com/content/wiki/Dev/CRX/Crx3/CQ5%20migration%20onto%20Oak/wiki:attachments/crx2oak-0.20.jar

Migration to TarMK:

1. Place the migration tool in the **<aem-install>** folder and run it with the following parameters:

```
java -jar crx2oak.jar [source repository] [target repository]
```

Where:

a. The source is the CRX repository that needs to be migrated. If you were running it from within the <aem-install> folder, this would be crx-quickstart/repository.
b. The target is the resulting TarMK repository to be created.

During the migration, the source repository should not be accessed by another client. The configuration is read from the repository.xml file within the source directory. The target repository is the filesystem path to the new TarMK repository folder, which will be created automatically if it does not exist.
The command should look like this:

```
java -jar crx2oak.jar crx-quickstart/repository oak-repository
```

In this example, the migrated TarMK repository will be created in a folder called oak-repository.

2. After the migration has finished, back up the AEM 5 DataStore and the newly created AEM 6.0 node store by copying them outside the installation folder. The folders that need to be copied are:
   a. crx-quickstart/repository/datastore
   b. oak-repository/segmentstore
3. Delete the entire crx-quickstart folder.
4. Re-create the crx-quickstart folder with additional configuration for setting the DataStore and node store for AEM 6.0:
   a. Create the **crx-quickstart/install** folder.
   b. In the install folder, create a file called: **org.apache.jackrabbit.oak.plugins.blob.datastore.FileDataStore.cfg**
   c. Edit the file and add these configuration options:

```
path=./crx-quickstart/repository/datastore
minRecordLength=4096
```

   d. In the same folder, create a file called: **org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStoreService.cfg**
   e. Add the following configuration option:

```
customBlobStore=true
```

5. Run the quickstart jar again.
6. Move the DataStore and segment store to the new installation. In order to do this, you need to:
   a. Delete all the contents of **crx-quickstart/repository**
   b. copy the **datastore** and **segmentstore** folders to **crx-quickstart/repository**
7. Start AEM.

3. Migrate to MongoDB from TarMK

Prerequisites:

a. Migration to MongoDB is currently supported only for versions that have been upgraded from AEM 5.6.1. For older versions of AEM 5 installed, you need to upgrade it to 5.6.1 first.
b. Make sure that MongoDB is installed and an instance of **mongod** is running. For more details, read the steps at http://docs.mongodb.org/manual/installation/.
c. Download the repository migration utility from USB or following link: https://wiki.day.com/content/wiki/Dev/CRX/Crx3/CQ5%20migration%20onto%20Oak/wiki:attachments/crx2oak-0.20.jar

Steps:

1. Copy the **AEM 6.0** jar into a separate folder.
2. In the new location, create the DataStore and node store configuration needed for the MongoDB migration:

- Create the **crx-quickstart/install** folder.
- In the **install** folder, create a file called **org.apache.jackrabbit.oak.plugins.blob.datastore.FileDataStore.cfg**.
- Edit the file and add these configuration options:

```
path=./crx-quickstart/repository/datastore minRecordLength=4096
```

- In the same folder, create a file called: **org.apache.jackrabbit.oak.plugins.document.DocumentNodeStoreService.cfg**
- Add the following configuration option:

```
customBlobStore=true
```

3. Start the **AEM 6.0** jar with a MongoMK backend by running:

```
java -jar aem-quickstart.jar -r crx3,crx3mongo  -Doak.mongo.uri=mongodb://remoteserver:27017/aem-author
```

where -r is the backend run mode. In this example, it will start with Mongo support. -Doak.mongo.uri is used to specify the location of the Mongo server and the database name.

NOTE: Both AEM and the repository migration tool use a MongoDB connection string to specify the location, port, and name of the database that needs to be used. For more information about the syntax, see Connection String URI Format at http://docs.mongodb.org/manual/reference/connection-string/.

4. Wait for the instance to start up, and then shut it down. After it has been shut down, prepare the migration by:
   - deleting the **crx-quickstart/repository** folder
   - deleting the **aem-author** Mongo database. This can be done from the Mongo shell by running:

```
use aem-author
db.dropDatabase()
```

   - moving the **crx-quickstart/repository/datastore** folder from the AEM 5 installation to the AEM 6.0 folder
5. Run the **crx2oak** upgrade tool and wait for the process to complete:

```
java -jar crx2oak.jar [source CRX 2 repository] mongodb://remoteserver:27017/aem-author
```

6. Restart AEM.
7. Test your new installation.
8. You can also start the instance with the following command:

```
java -Xms512m -Xmx4096m -XX:MaxPermSize=256m -Djava.awt.headless=true -jar cq-quickstart-6.0.jar -nobrowser -nofork -r crx3,crx3mongo -p4502 -r author -Doak.mongo.db=OakAuthor -Doak.mongo.uri=mongodb://remoteserver:27017  -use-control-port
```

-r is the backend run mode.

WARNING: Any custom repository configurations (such as LDAP integration) will have to be re-created for OAK after the upgrade.

Install Dispatcher on IIS

In this section, you will install Dispatcher into an IIS web server.

To successfully complete and understand these instructions, you will need:

- An IIS web server, up and running
- A running AEM Author instance
- A running AEM Publish Instance

Use Adobe PackageShare (https://www.adobeaemcloud.com/content/companies/public/adobe/dispatcher/dispatcher.html) to obtain the latest Dispatcher installation file for your operating system and web server. Dispatcher release numbers are independent of the AEM release numbers and are compatible with AEM 5.x and Adobe CQ 5.x releases.

The following file naming convention is used:

dispatcher-<web-server>-<operating-system>-<dispatcher-version-number>.<file-format>

For example, the dispatcher-apache2.2-linux-i686-4.0.6.tgz file contains Dispatcher version 4.0.6 for an Apache 2.2 web server that runs on Linux i686 and is packaged using the tar format.

The following table lists the web server identifier that is used in file names for each web server:

Web server	Installation kit
Apache 2.4	dispatcher-apache2.4-<other parameters>
Apache 2.2	dispatcher-apache2.2-<other parameters>
Microsoft IIS 7.5, 8, 8.5	dispatcher-iis-<other parameters>
Sun Java Web Server/iPlanet	dispatcher-ns-<other parameters>

Make sure IIS is up and running in your system; if not, refer to the following documentation for installation instructions:

http://msdn.microsoft.com/en-us/library/ms143748%28SQL.90%29.aspx

1. Required IIS components:
   IIS versions 7.5, 8.0, and 8.5 require that the following IIS component be installed:
   a. ISAPI Extensions

Also, you must add the web server (IIS) role. Use Server Manager to add the role and components.

i. The required archive for Microsoft Internet Information System(IIS) is: **dispatcher-iis-<operating-system>-<dispatcher-release-number>.zip**

The zip file contains the following files:

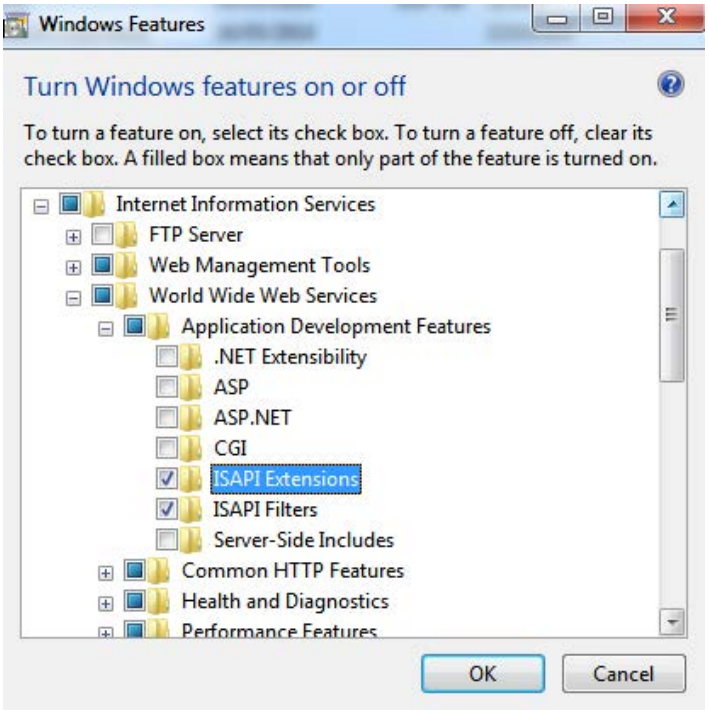
File	Description
disp_iis.dll	The Dispatcher dynamic link library file.
disp_iis.ini	Configuration file for the IIS. This example can be updated with your requirements. NOTE: The .ini file must have the same name-root as the .dll file.
dispatcher.any	An example configuration file for Dispatcher.
author_dispatcher.any	An example configuration file for Dispatcher working with the author instance.
release-notes.txt	Installation instructions, a list of fixed issues in the current and previous releases, and other last-minute information. **Note:** Read this file before installing.

b. Use the following procedure to copy Dispatcher files to the correct location.

Use Windows Explorer to create the **<IIS_INSTALLDIR>/Scripts** directory, for example, **C:\inetpub\Scripts**.

c. Extract the following files from Dispatcher package into this **Scripts** directory:
   i. disp_iis.dll
   ii. disp_iis.ini
   iii. One of the following files depending on if Dispatcher is working with an AEM author instance or Publish Instance:
      . Author instance: author_dispatcher.any
      . Publish Instance: **dispatcher.any**

Also, remember to check that **Role Services** are installed:

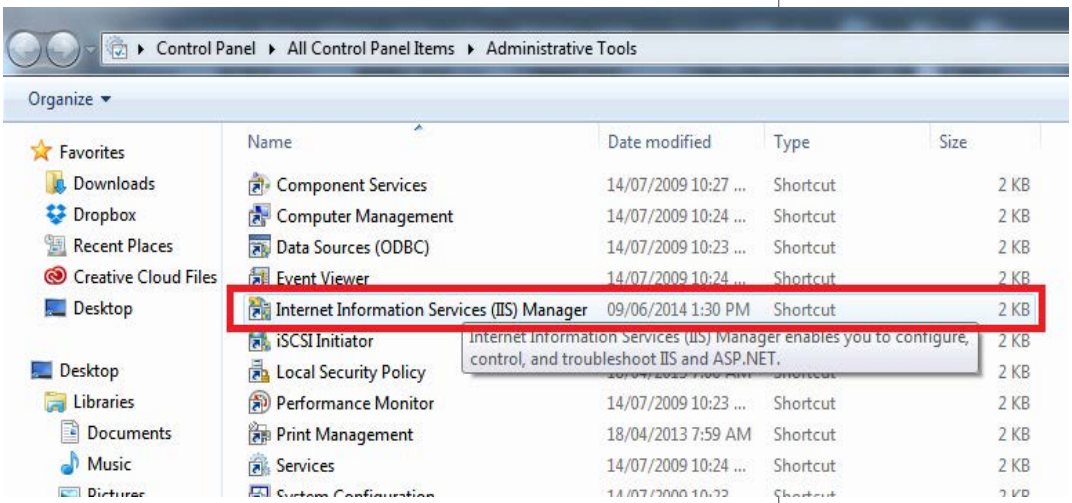


Feature Window of Microsoft IIS

2. Unzip the latest Dispatcher build, appropriate for your operating system, to a temporary directory. The Dispatcher files are located on the memory stick under **/distribution/dispatcher**.

3. Add Dispatcher to the list of available ISAPI filters (by adding the DLL to the IIS) using the following steps:
   a. Extract **disp_iis.dll, disp_iis.ini, disp_iss.pdb** and **dispatcher.any** into the executable directory of the selected website under IIS, for example, **<IIS_INSTALLDIR>/scripts**.
   b. Configure the Virtual Directory.
   c. Open **Administrative Tools**, and then select **IIS Manager**.

NOTE: The ISS_INSTALLDIR is normally the C:\Inetpub folder.



d. Select your site in the tree, and then from the context menu (usually right-click), select **Add Virtual Directory.**
e. Enter the alias **scripts** and the physical path of the directory created above (**C:\Inetpub\scripts**).

4. Register the ISAPI **filter**:
   a. Select **Features View**.
   b. Open the feature **ISAPI Filters** in the view.
   c. Click **Add**, and enter the following settings:
      Filter name: AEM
      Executable: The path to **disp_iis.dll** (C:\Inetpub\scripts)
   d. Click **OK** to save.

5. Register the ISAPI **handler**:
   a. Open the feature **Handler Mappings** in the view.
   b. Click **Add Script Map**, and enter the following settings:
   c. **Request Path:** /scripts/disp_iss.dll
   d. **Executable:** the path to disp_iis.dll (C:\Inetpub\scripts)
   e. **Name**: AEM
   f. Click **OK** to save.
   g. Open the feature **Configuration Editor**.
   h. Select **system.webServer\handlers** from the section drop-down list.
   i. Select the first row (**Collection** at the top), and then click the **three dots** button (at the far right).
   j. The Collection Editor appears; select the handler, **AEM**.
   k. Change the value of the **allowPathInfo** flag to **true**.
   l. Close the Collection Editor and click **Apply**.


Refer to Appendix-B under the section 'Dispatcher Installation on a Different Version of Microsoft IIS' for different versions of IIS.

6. Register the json extension:
   a. Open the feature **MIME Types** in the view.
   b. Click **Add**, and enter the following settings:
      File name extension: json
      MIME type: application/json
   c. Click **OK** to save

7. Place the (example) configuration file **disp_iis.ini** into the same directory as the DLL **disp_iis.dll.** The basic format of the .ini file is as follows:

```
;This is the initial configuration file for the ISAPI filter DLL. Its name (without
; extension) must match the base name of the ISAPI filter DLL.
[main]
; scriptpath is the URI pointing to the ISAPI extension DLL
scriptpath=/scripts/disp_iis.dll
; configpath is the physical filesystem path of the dispatcher.any configuration file
configpath=C:\Inetpub\scripts\dispatcher.any
; loglevel is the logging level, values allowed lie between 0 (NONE) and 3 (DEBUG)
loglevel=3
; servervariables, if set to 1, forwards IIS server variables as request headers to
; the remote instance.
servervariables=0
; replaceauthorization,
; if set to 1, replaces any header named "Authorization" other than
; "Basic" with its "Basic <IIS:LOGON_USER>" equivalent.
replaceauthorization=0
; declineroot, if set to 1, declines requests to "/" and passes them on to the next
; filter or IIS, it this is the last filter in the filter chain
;declineroot=0
```

Congratulations! You successfully integrated the AEM Dispatcher with the IIS web server.

NOTE: It is recommended to set the log level to 3 during installation and testing, and then revert to 0 when running in a production environment.

NOTE: Before you start using the AEM Dispatcher, you must configure Dispatcher (See the chapter titled 'Configuring the Dispatcher').