BEA AquaLogic Commerce Services

Deployment Guide

Version 6.0
March 2008

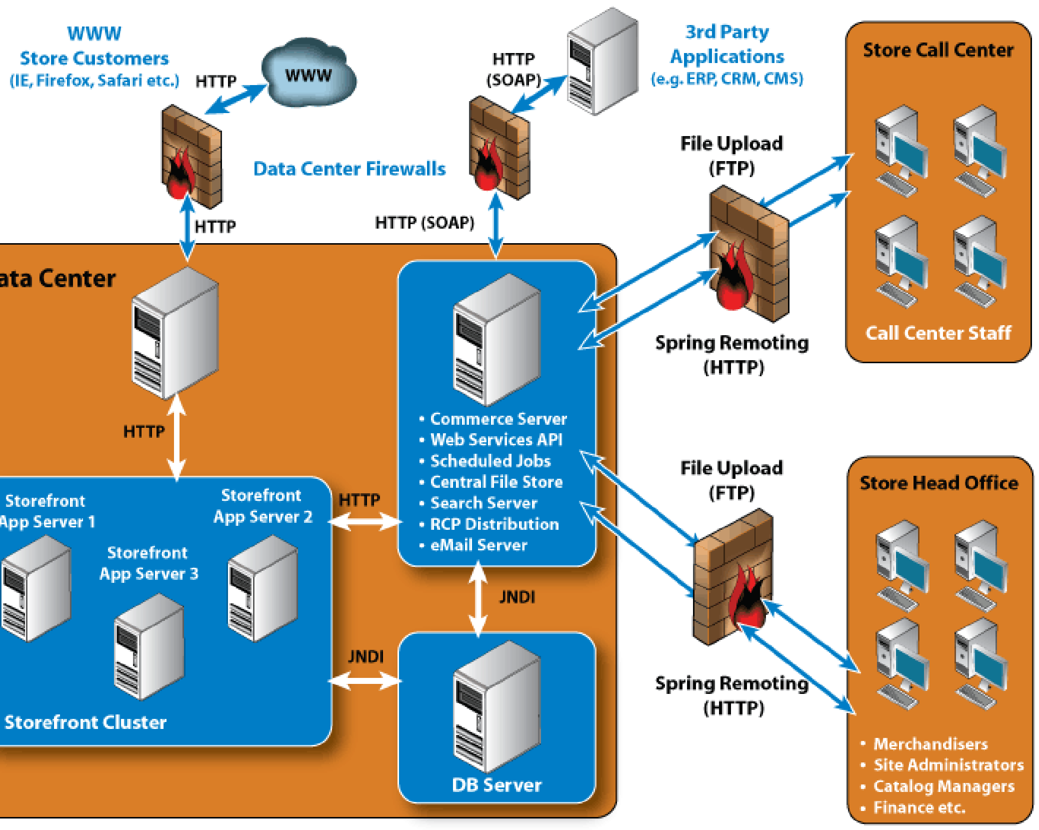
Contents

Overview

This guide will help you deploy AquaLogic Commerce Services on a supported server platform and configure it to meet your needs. It assumes that you have a binary distribution for the application server upon which you intend to deploy the application.

The possible distributions and overview of steps are below, and assume that you want to deploy the sample SNAPITUP storefront. Note that the architecture of AquaLogic Commerce Services currently requires that it be deployed as an exploded EAR.

A high-level diagram of the AquaLogic Commerce Services platform can be found below:



The general steps described in this guide are:

- Creating a database
- Deploying AquaLogic Commerce Services
- Configuring AquaLogic Commerce Services
- Load balancing & clustering (optional)

Creating a Database

AquaLogic Commerce Services is shipped with a sample embedded Derby database configured with fictional SNAPITUP store data for demonstration purposes. For production and development, a production grade database should be used. This section explains how to create an AquaLogic Commerce Services database schema using one of the following supported databases:

- Microsoft SQL Server 2005
- MySQL 5.0.x
- Oracle 10g

Microsoft SQL Server 2005

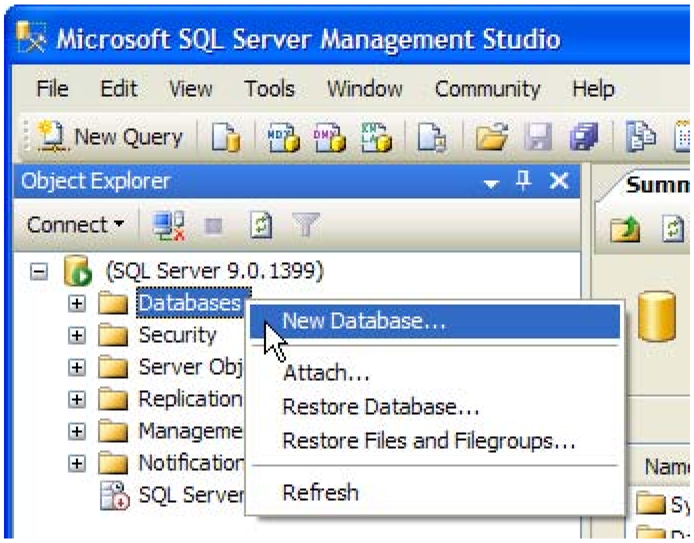
Note: The following instructions assume that Microsoft SQL Server 2005 is running, and that the SQL Server Management Studio is installed.

Connect to Microsoft SQL Server

1. Launch the SQL Server Management Studio by clicking Start > Microsoft SQL Server 2005 > SQL Server Management Studio
2. Enter the login information for your SQL Server and click **Connect**.

Create ALCS Database

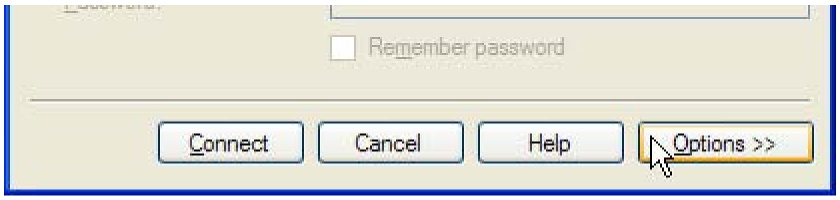
1. Right-click the Databases node in the Object Explorer, and click **New Database...**



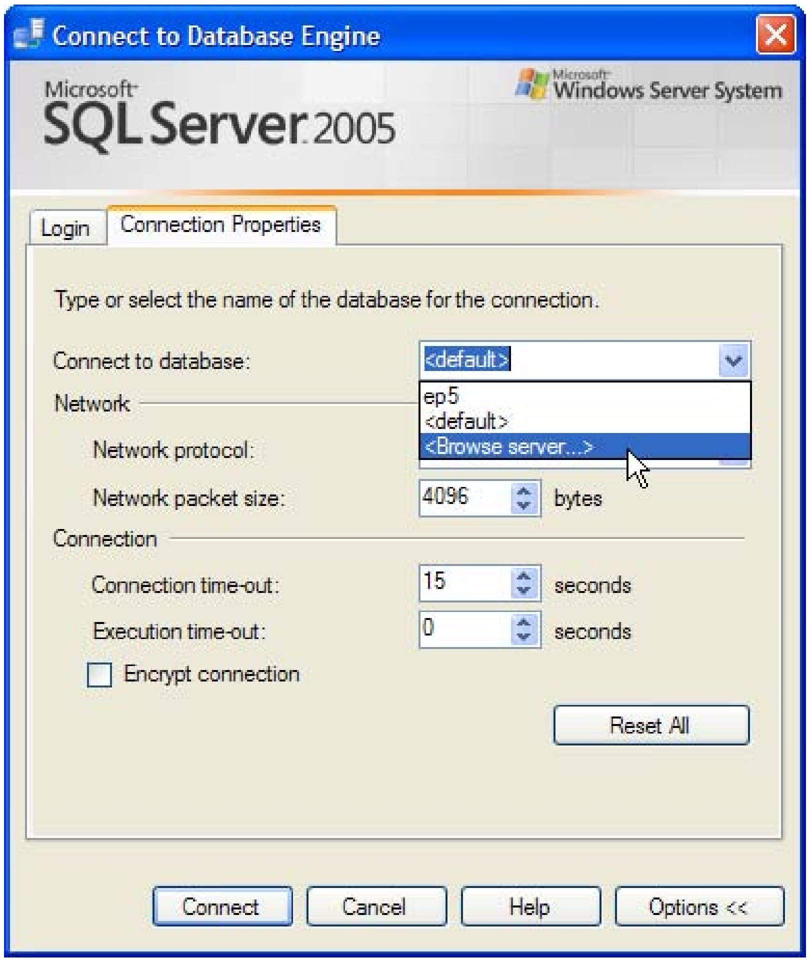
2. Enter **alcs** in the **Database name** field, and click **OK** to create the database. It may take a few seconds

Populate Database

1. On the **File** menu point to **Open**, and then click on **File...**
2. Navigate to the **<BEA_HOME>/commerce/database-scripts/mssql** folder within the CD or download location of AquaLogic Commerce Services (where <BEA_HOME> refers to the path where AquaLogic Commerce Services is installed).
3. Select **schema.sql** and click **Open**
4. In the **Connect to Database Engine** dialog click on the **Options >>** button that is in the bottom right corner.



5. Click on the dropdown next to **Connect to database** and select **<Browse server...>**.



Connect to Database Engine
Microsoft SQL Server 2005
Microsoft Windows Server System
Login
Connection Properties

Type or select the name of the database for the connection.

Connect to database: <default>

ep5
<default>
<Browse server...>

Network

Network protocol:

Network packet size: 4096 bytes

Connection

Connection time-out: 15 seconds

Execution time-out: 0 seconds

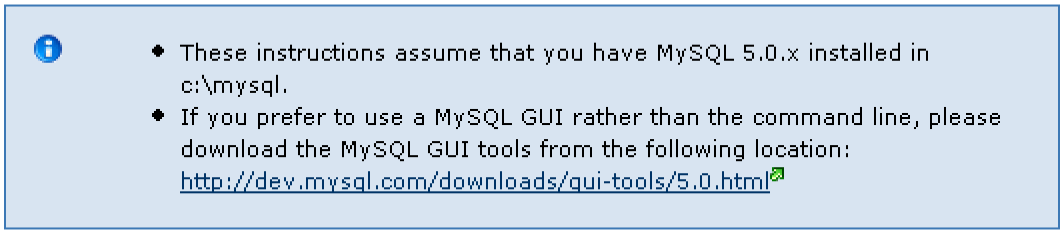
Encrypt connection

Reset All
Connect
Cancel
Help
Options <<

6. In the Browse for databases dialog click Yes

7. Navigate to the database you created in the previous section, and click OK
8. Click Connect
9. On the Query menu click on Execute. This may take a few seconds
10. Repeat steps 1 - 9 for the rest of the .sql files in the following order:
    - base-insert.sql
    - snapitup-insert.sql (Optional, deploys the AquaLogic Commerce Services Demo Store "Snap It Up" on your database)
11. When you are finished the above steps you may exit the SQL Server Management Studio
12. If you chose to run the snapitup-insert.sql script you will also need to import the SnapItUp product catalog data

MySQL 5.0.x



- These instructions assume that you have MySQL 5.0.x installed in c:\mysql.
- If you prefer to use a MySQL GUI rather than the command line, please download the MySQL GUI tools from the following location: http://dev.mysql.com/downloads/gui-tools/5.0.html

If you have not configured your MySQL server to start automatically, follow the steps below to start your server.

1. Ensure that your MySQL service is configured to start automatically (on Windows or Linux), or start your database server as follows (Windows only):

    a. Open a DOS command window by clicking: Start->Run
    b. Type: cmd and then press ENTER
    c. Change folder to: c:\mysql\bin
    d. Type: mysqld -nt and press ENTER to start the database.
    e. Open another DOS command window by clicking: Start->Run
    f. Type: cmd and then press ENTER
    g. Change folder to: c:\mysql\bin

2. Repeat steps (a) to (c) above to open another DOS command window, and type the following to enter mysql console:

```
mysql -u root
```

   If you configured the root user to use a password, type

```
mysql -u root -p
```

   Type the password for the root user when prompted.

3. Create a database called ALCS.

```
mysql> create database ALCS character set utf8;
```

> If you choose to use a database name other than ALCS, please make sure this value is configured correctly when setting up your data source. See the **JNDI/JDBC Configuration** section below.

4. Create a user account.

   If the MySQL server and the web application server run on different hosts, create a user account and grant privileges to the host which runs the web application server with the following command:

```
mysql> grant all privileges on ALCS.* to username@'ipaddress';
```

   If your database is on a different machine than the web application server, you may want to specify a password by adding identified by 'password' to your grant command.

   Commit the permission changes:

```
mysql> flush privileges;
```

5. Change database to ALCS:

```
mysql> use ALCS;
```

6. Copy the sql files from <WL_HOME>/commerce/database-scripts/mysql in your CD or installation download directory to c:\mysql\bin.
7. Create a blank ALCS database:

```
mysql> source schema.sql;
mysql> source base-insert.sql;
```

8. (Optional, deploys the AquaLogic Commerce Services Demo Store "Snap It Up" on your database)

```
mysql> source snapitup-insert.sql;
```

9. If you chose to run the snapitup-insert.sql script you will also need to import the SnapItUp product catalog data.

Oracle 10g

In the instructions below, <ORACLE_BASE> refers to the Oracle home directory on your system, e.g. C:\Oracle\product\10.2.0\db_1

> ⚠ When installing Oracle 10g on a Windows system as a domain user as opposed to a local administrator you must set the SQLNET.AUTHENTICATION_SERVICES configuration value to NONE in the <ORACLE_BASE>/NETWORK/ADMIN/sqlnet.ora file.
>
> ⚠ In order to install Oracle on a Windows system with a dynamically assigned IP address you must install the Microsoft Loopback Adapter: http://support.microsoft.com/kb/839013

The following instructions assume that Oracle is already installed and running.

Creating the ALCS Database

1. Launch the Oracle Database Configuration Assistant via the start menu by clicking Start » All Programs » Oracle - OraDb10g_home1 » Configuration and Migration Tools » Database Configuration Assistant.
2. Select Create a Database, and click Next
3. Select General Purpose, and click Next
4. Enter alcs as the Global Database Name, and the SID
5. Click Next on step 4
6. Enter a password, and click Next
7. Select File System, and click Next
8. Select Use Database File Locations from Template, and click Next
9. Click Next on step 8
10. Ensure that Sample Schemas is unchecked, and click Next
11. On the Character Sets tab select Choose from the list of character sets, and select UTF8 - Unicode 3.0 UTF-8 Universal character set, CESU-8 Compliant, and click Finish
12. Click OK on the confirmation dialog, and the database creation process will start. It may take a few minutes to complete
13. Once you've created the database

Creating the ALCS Schema

1. Click **Start** » **Run**
2. Enter **cmd** into the **Open** field, and click **OK** to launch the Windows command line
3. Navigate to the **<WL_HOME>/commerce/database-scripts/oracle** folder within the CD or download location of AquaLogic Commerce Services
4. Type **sqlplus** into the command prompt, and press Enter

5. Type **@schema.sql into the** SQL>* prompt, and press Enter
6. Repeat step 5 for the rest of the .sql files in the following order:
    - base-insert.sql
    - snapitup-insert.sql (Optional, deploys the AquaLogic Commerce Services Demo Store "Snap It Up" on your database)
7. When you are finished type quit to exit SQL*Plus
8. If you chose to run the snapitup-insert.sql script you will also need to import the SnapItUp product catalog data

Running SnapItUp Import Jobs (optional)

This section is for users who wish to import the SnapItUp product catalog data from the CSV files provided with AquaLogic Commerce Services.

1. Log into the Commerce Manager Client
2. Click on "Activity" from the menu
3. Click on "Catalog Management" to go to Catalog Management perspective.
4. Choose "View Import Jobs" from the tool bar
5. Choose the first import job from the picklist called "01-SnapItUp"
6. Click the icon at the end of the "CSV Template File", and select 'snapitup.csv' under assets/import folder
7. Choose "Next > "
8. Click "Finish" and wait for the import job to finish.

Repeat steps 5 through 8 for each existing import job in the picklist, being sure to import them in the order in which they are numbered (01, 02, 03, etc). In addition to importing the catalog for the SnapItUp store, its inventory must be imported or the imported products will not have any inventory and will not be displayed in the storefront.

1. Click on "Activity" from the menu.
2. Click "Shipping/Receiving" to go to the Warehouse perspective.
3. Choose "View warehouse import jobs" from the tool bar.
4. Click the "Inventory" job from the pick list.
5. Click the icon at the end of the "CSV Template File", and select the corresponding inventory csv file for each category (e.g. accessories-inventory.csv under the assets/import folder).
6. Choose "Next > ".
7. Click "Finish" and wait for the import job to finish.
8. Repeat steps 5 through 8 for each inventory CSV file to be imported.

Deploying AquaLogic Commerce Services Application in a new Domain

This section explains how to deploy AquaLogic Commerce Services applications on BEA WebLogic Server 9.2 or10.0. If you'd like more information on where to find installed files and the sample domain installed by default with AquaLogic Commerce Services installer, refer to the *AquaLogic Commerce Services Getting Started Guide*

BEA WebLogic 9.2 or 10.0

These instructions are tailored to WebLogic Server 10.2 and are roughly equivalent to the instructions for WebLogic 9.2MP2 unless otherwise specified.

Throughout this chapter, <WL_HOME> refers to the path where BEA WebLogic Server is installed, such as c:\bea

The date and time of all application servers should be synchronized with the date and time of your database server to prevent confusing creation/last-modified timestamps from being created in the database. This issue arises from the fact that some code uses the database server date/time and other code uses the application server date/time.

WebLogic-specific WAR files

Because AquaLogic Commerce Services applications access files directly from the file system they must be deployed in an exploded form. The Web Services application must be deployed inside an EAR to take advantage of classloader filtering, which eliminates conflicts with XML libraries (QName.class, specifically) that are included in WebLogic's own library files.

Quick Start

Assumptions:

1. You have a database already installed and created.
2. You have the correct license file installed,

Deployment Steps:

1. Install WebLogic Server.
2. Create a new WebLogic Domain
3. Download and install Java Advanced Imaging (JAI) to the JDK that WebLogic is using.

4. Copy the database JDBC driver into your domain's lib directory
5. Make a copy of the exploded EAR application from the installation directory. i.e. commerce_6.0\samples\commerce\commerceApp
6. Configure the web applications through the configuration files (commerce-config.xml, quartz.xml)
7. Edit WebLogic Server's startup scripts (point to the correct JDK, JVM memory usage)
8. Follow the library conflict-resolution instructions for your version of WebLogic.
9. Start WebLogic Server and go through the deployment instructions, including setting up JNDI.

Detailed Deployment Instructions

Create a New Domain

1. Click Start > All Programs > BEA Products > Tools > Configuration Wizard.

> For Linux, you can run <BEA_HOME>/common/bin/config.sh.
> You must have an X server configured to use the GUI configuration tool.

2. Select 'Create A new WebLogic domain' in the window that appears. Click the 'Next' button in the lower right corner.
3. Select 'Generate a domain configured automatically to support the following BEA Products:'. Click 'Next'.
4. Enter a user name and user password. For this example, we will use *weblogic* as the User name and *weblogic* as the password. Write down this information, as you will need it later. Click 'Next'.
5. Select the BEA supplied 'Sun SDK 1.5.0' or point the application to the Java Development Kit of your choice. Click 'Next'.

> **JAI must be installed on the selected JDK.**
>
> Remember that whichever JDK you select must have Java Advanced Imaging libraries installed.

6. In the next screen, leave 'No' selected and click 'Next'.
7. Enter the name and location for the domain and click 'Create'. For this example, we will use *alcs* as the name and <BEA_HOME>\user_projects\domains as the location. WebLogic will create a directory <BEA_HOME>\user_projects\domains\alcs.
8. The wizard will now create a new domain. Click 'Done' when the wizard finishes to close the window.

Unpack the .war files

During operation AquaLogic Commerce Services directly manipulates some files that are contained in the WAR; therefore, the applications can only be deployed as exploded WARs.

1. Copy the *.war files from your CD or download location to the directory in which you would like them deployed.

For our example, we will use C:\WarHome\storefront for Storefront, C:\WarHome\cmserver for CommerceManager, C:\WarHome\searchserver for the search server, and C:\WarHome\webservices for Web Services.

2. Unpack each .war file:

a. On Windows:

a) Click Start > Run.

b) Type cmd in the the window that appears. Click OK.

c) Navigate to the directory where the alcssf.war file was copied to. For our example, C:\War Home\alcssf

d) Unpack the war file with the jar utility by calling

```
jar -xvf alcssf.war
```

Set Up the Domain

1. To correctly set up a JDBC connection, the JDBC driver must be on WebLogic Server's CLASSPATH. To do this, copy the JDBC driver file for your database into the <DOMAIN_HOME>\lib directory (see the JNDI and JDBC Configuration section for more information).

2. You may get compatibility exceptions on *.QName when starting WebLogic Server with some releases of JDK 1.5. To correct this problem, you will need to edit <DOMAIN_HOME>\bin\setDomainEnv.cmd and add the following lines:

On Windows:

```
set JAVA_OPTIONS="%JAVA_OPTIONS%
  -Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0"
```

On Linux:

```
JAVA_OPTIONS="${JAVA_OPTIONS} -
Dcom.sun.xml.namespace.QName.useCompatibleSerialVersionUID=1.0"
export JAVA_OPTIONS
```

3. Depending on how many web applications you plan to run in the same instance of WebLogic Server, you may find that different memory settings are required to avoid out of memory exceptions in the JVM. As a recommended starting point, the maximum heap size should be set to 1024MB ("-Xmx1024m") or higher and the maximum perm space size to 256MB ("-XX:MaxPermSize=256m"). These settings can be changed in <BEA_HOME>\user_projects\domains\alcs\bin\setDomainEnv.cmd or setDomain.sh (the MEM_ARGS and MEM_DEV_ARGS settings). Alternatively, if you

specify an environment variable (USER_MEM_ARGS = -Xmx1024m -XX:MaxPermSize=256m) it should override any settings in the script.

4. For promotions to work correctly, the jar stringtemplate-stringtemplate-2.3b6.jar must be inserted into Java's classpath before weblogic.jar. To accomplish this, copy the jar of that name from the WEB-INF/lib directory of the unpacked cmserver.war file into <DOMAIN_HOME>/lib and edit the PRE_CLASSPATH variable in setDomainEnv.sh (setDomainEnv.cmd on Windows) as follows:

   On Windows:

   ```
   PRE_CLASSPATH=%DOMAIN_HOME%\lib\stringtemplate-stringtemplate
   2.3b6.jar
   ```

   On Linux:

   ```
   PRE_CLASSPATH="${DOMAIN_HOME}/lib/stringtemplate-stringtemplate-2.3b6.jar"
   ```

Configure the Web Applications

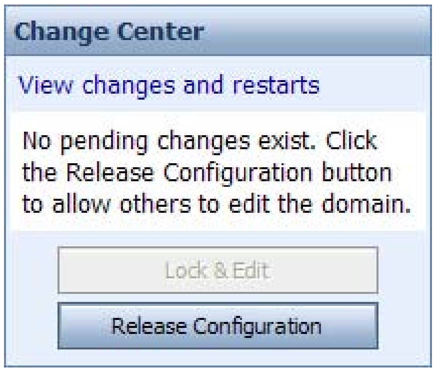
The commerce-config.xml file must be updated in all AquaLogic Commerce Services web applications (searchserver, cmserver, storefront, webservices). Where applicable, the quartz.xml files should also be updated.

Deployment with WebLogic Administrator Console

1. Start the WebLogic Server by double-clicking <BEA_HOME>\user_projects\domains\alcs\startWebLogic.cmd
2. Open the Administration Console.
   a. Open a browser window.
   b. Browse to http://localhost:7001/console.
   c. Type the username and password entered during step 4 of 'Create a New Domain'. For our example, it will be weblogic / weblogic.

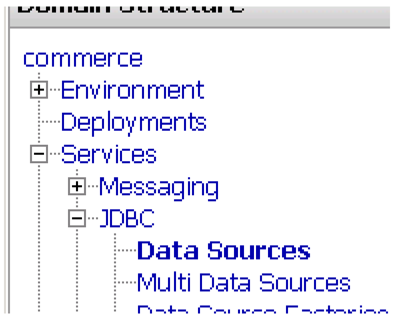
Set Up a JNDI Data Source

1. On the left hand side, click the 'Lock and Edit' button under the Change Center.

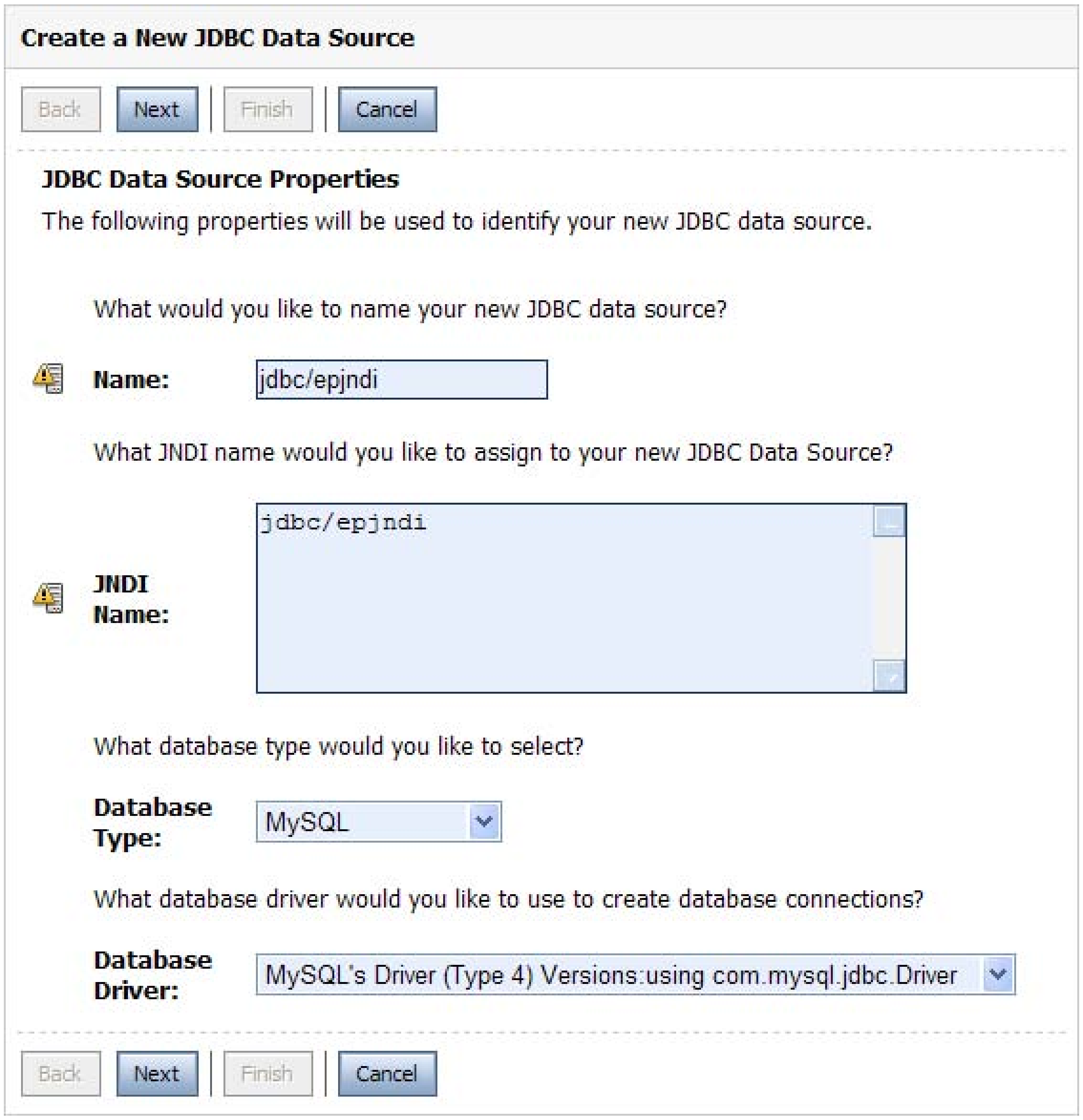


No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

2. Expand the alcs domain under the 'Domain Structure' Tree. Select Services > JDBC > Data Sources.



3. On the right hand pane, click the 'New' button under Data Sources.
4. On the next page titled 'JDBC Data Source Properties', enter the following information:



JDBC Data Source Properties

The following properties will be used to identify your new JDBC data source.

What would you like to name your new JDBC data source?

Name: jdbc/epjndi

What JNDI name would you like to assign to your new JDBC Data Source?

JNDI Name: jdbc/epjndi

What database type would you like to select?

Database Type: MySQL

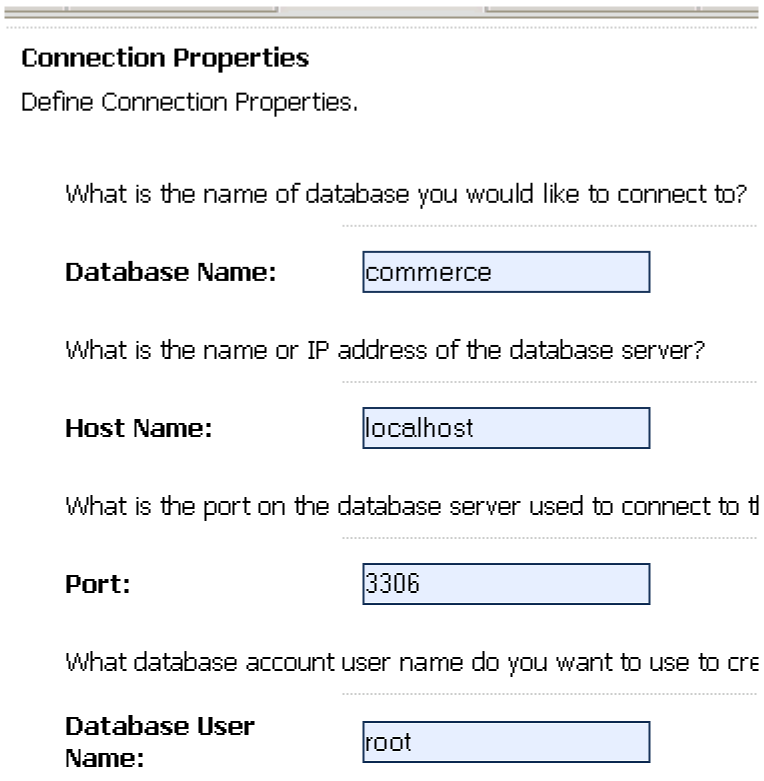
What database driver would you like to use to create database connections?

Database Driver: MySQL's Driver (Type 4) Versions:using com.mysql.jdbc.Driver

- **Name:** jdbc/epjndi
- **JNDI Name:** jdbc/epjndi
- **Database Type:** Select your database type
- **Database Driver:** Select the database driver copied in the Deploy JDBC Jar File stage. (See 2 - JNDI and JDBC Configuration for the correct Driver name)

Click 'Next'.

5. Click Next on the Transaction Options screen.
6. Enter your database properties on the Connection Properties screen. Click Next.



Connection Properties

Define Connection Properties.

What is the name of database you would like to connect to?

Database Name: commerce

What is the name or IP address of the database server?

Host Name: localhost

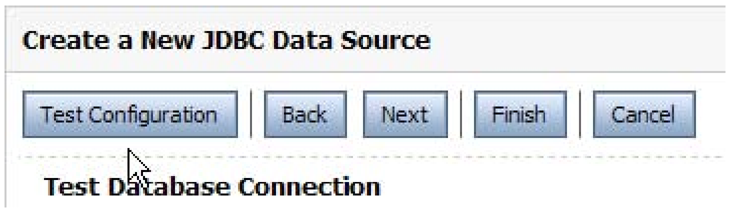
What is the port on the database server used to connect to t

Port: 3306

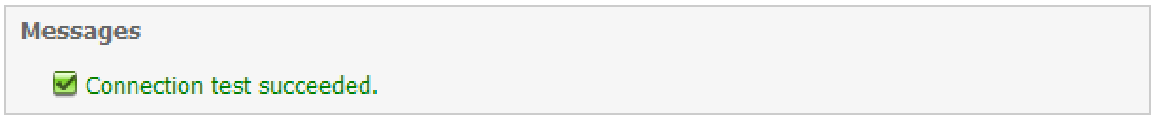
What database account user name do you want to use to cre

Database User Name: root

7. In the 'Test Database Connection' screen, click on the 'Test Configuration' button.



If your connection information is correct, you should see a success message. Fix any errors reported otherwise.

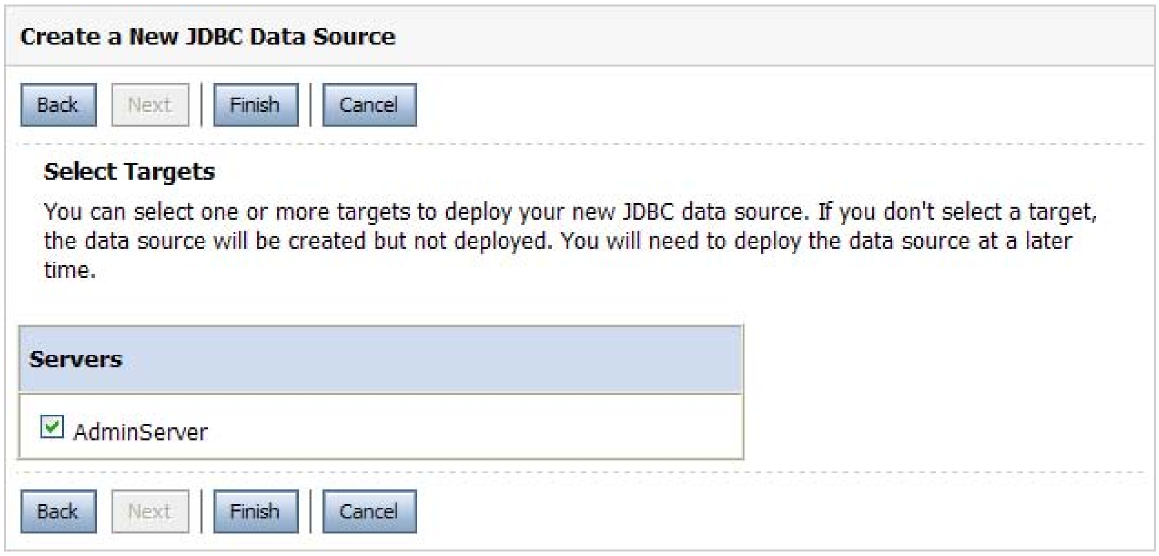


Messages

✓ Connection test succeeded.

Click 'Next'.

8. In the 'Select Targets' screen, click on the checkbox next to 'AdminServer'. Click 'Next'.

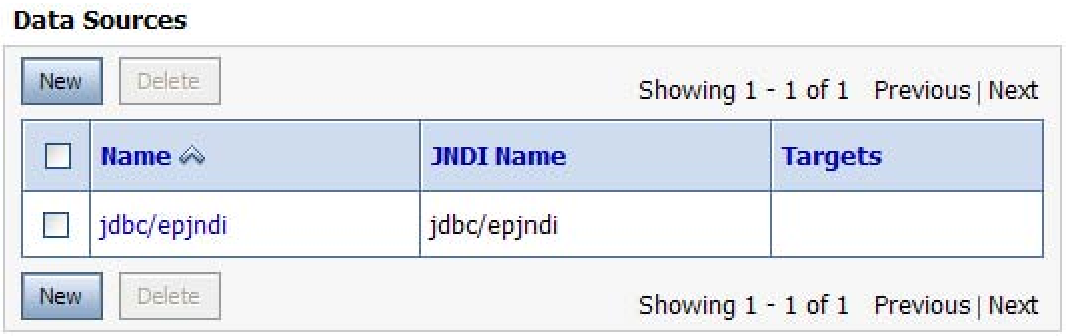

Create a New JDBC Data Source

Select Targets

You can select one or more targets to deploy your new JDBC data source. If you don't select a target, the data source will be created but not deployed. You will need to deploy the data source at a later time.

Servers
✓ AdminServer

9. Click 'Finish' to save the configuration. The new data source should be visible in the Data Sources window.

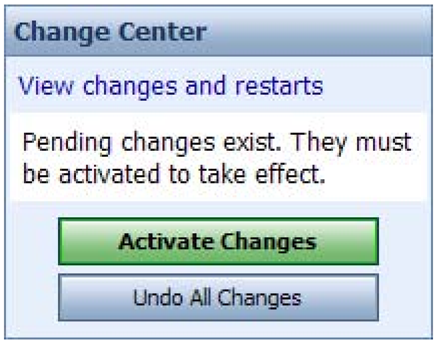

Data Sources

New
Delete
Showing 1 - 1 of 1 Previous | Next

☐	Name	JNDI Name	Targets
☐	jdbc/epjndi	jdbc/epjndi	

New
Delete
Showing 1 - 1 of 1 Previous | Next

10. On the left hand side, click the 'Activate Changes' button under the Change Center to commit these changes. You have completed setting up your JDBC Connection.


Change Center

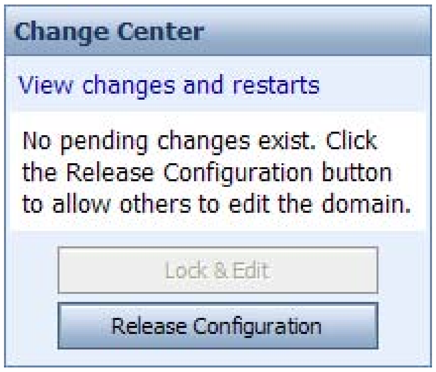
View changes and restarts

Pending changes exist. They must be activated to take effect.

You might have to restart your Weblogic server before you can successfully commit changes when setting up a JDBC Connection.

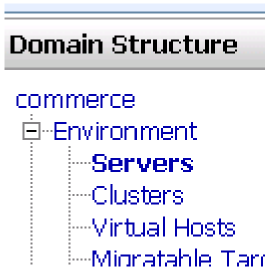
Configure SSL (Optional)

1. On the left hand side, click the 'Lock and Edit' button under the Change Center.

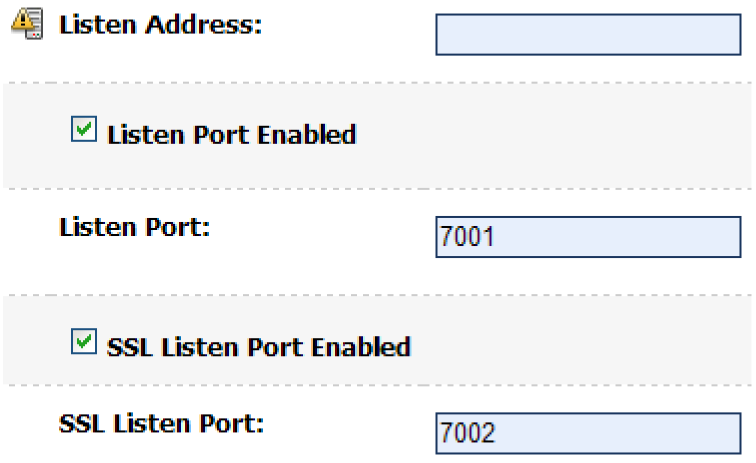


No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

2. Expand the alcs domain under the 'Domain Structure' Tree. Expand 'Environment' > Servers



3. Click on the name 'AdminServer(admin)' in the Server list.
4. In the General Configurations window, check the 'SSL Listen Port Enabled' box.



Listen Address:

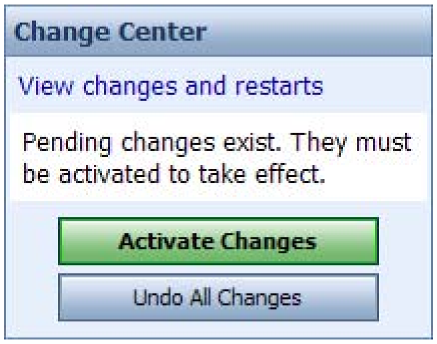
☑ Listen Port Enabled

Listen Port: 7001

☑ SSL Listen Port Enabled

SSL Listen Port: 7002

5. Click 'Save'.
6. On the left hand side, click the 'Activate Changes' button under the Change Center to commit these changes. You have finished configuring SSL.

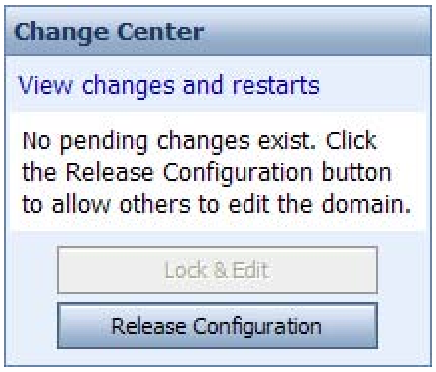

Change Center

View changes and restarts

Pending changes exist. They must be activated to take effect.

Deploy AquaLogic Commerce Services

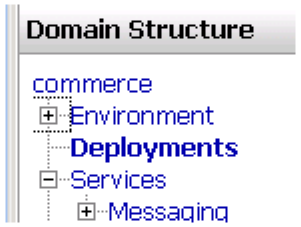
1. On the left hand side, click the 'Lock and Edit' button under the Change Center.


Change Center

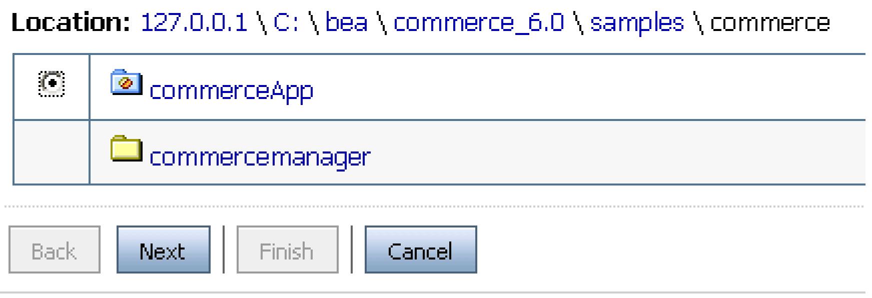
View changes and restarts

No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

2. Expand the alcs domain under the 'Domain Structure' Tree. Select 'Deployments'.

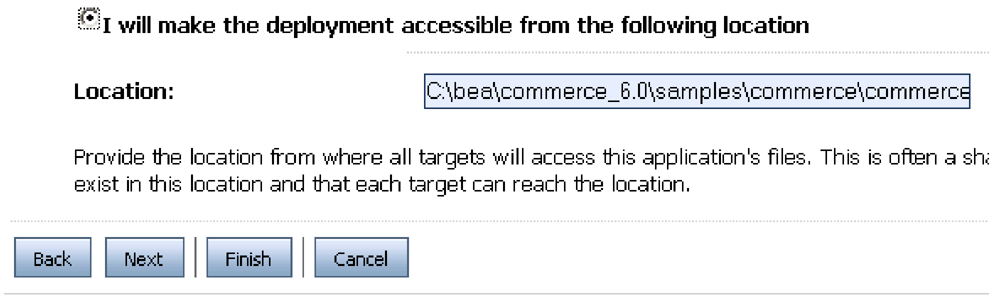


3. Click the 'Install' button in the Deployments pane on the right.
4. Navigate to the source of your exploded ear directory. Click 'Next'.


Location: 127.0.0.1 \ C: \ bea \ commerce_6.0 \ samples \ commerce

5. In the 'Choose Targeting Type' screen, select 'Install this deployment as an application'. Click 'Next'.

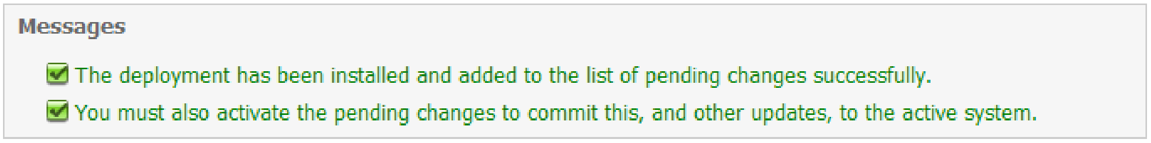
6. In the 'Optional Settings' screen, make sure the 'I will make the deployment accessible from the following location' option is selected for the 'Source accessibility' option. Click 'Finish'.


[✓ I will make the deployment accessible from the following location]

Location: C:\bea\commerce_6.0\samples\commerce\commerce

Provide the location from where all targets will access this application's files. This is often a sh
exist in this location and that each target can reach the location.

7. You should see a success message, and your web application should appear under the 'Deployments' pane.

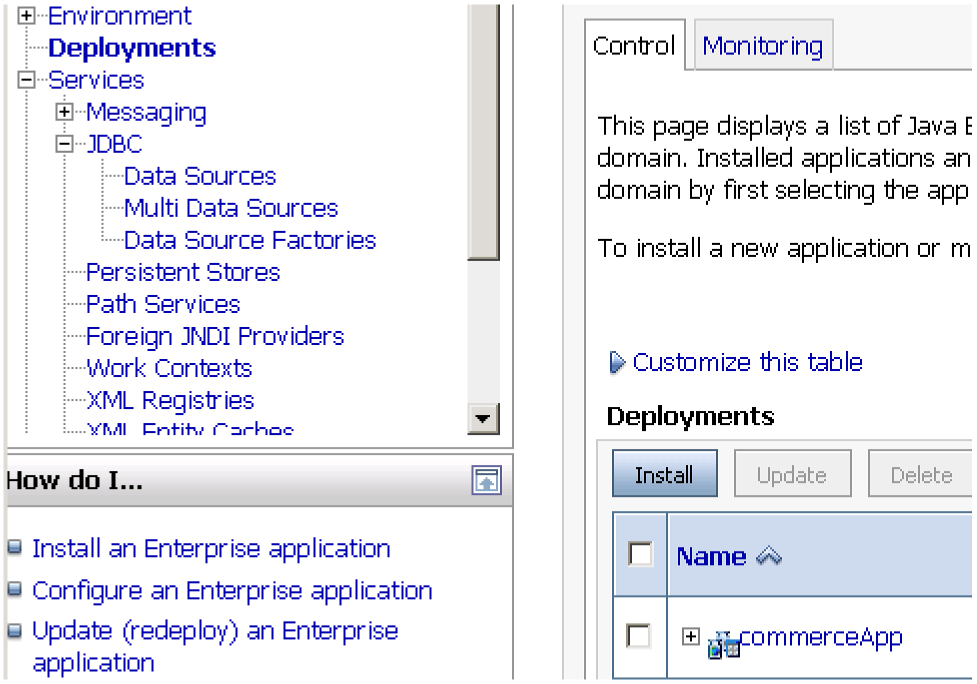

Messages

✓ The deployment has been installed and added to the list of pending changes successfully.

✓ You must also activate the pending changes to commit this, and other updates, to the active system.

8. Repeat steps 3-7 for C:\War Home\alcssf.

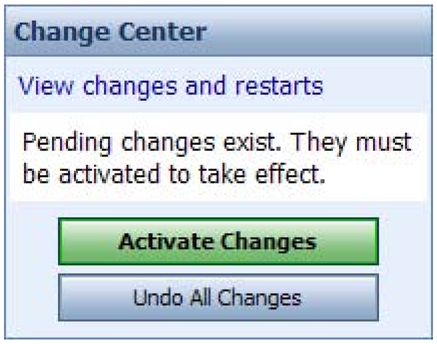
9. You should see both web applications available under the 'Deployments' pane.



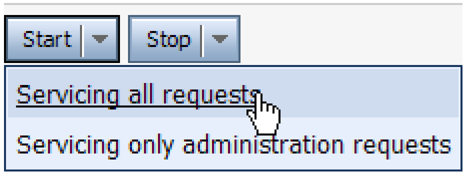
This page displays a list of Java E
domain. Installed applications an
domain by first selecting the app

To install a new application or m

10. On the left hand side, click the 'Activate Changes' button under the Change Center to commit these changes.


Change Center
View changes and restarts
Pending changes exist. They must be activated to take effect.
Activate Changes
Undo All Changes

11. Select the checkbox next to each application. Click on the 'Start' button and select 'Servicing all requests'. Click 'OK' on the confirmation screen to start your application. You have finished deploying AquaLogic Commerce Services.



WebLogic library compatibility problems and solutions

AquaLogic Commerce Services includes a few open source framework jar files for web services and utility functions, some of which are already included in WebLogic Server 9.2 and 10.
To avoid version conflicts which may result in class incompatibility errors on deployment, we need to ensure that the AquaLogic Commerce Services domain is using the correct version of classes/jars.

WebLogic Server 9.2 instructions

An older web service and xml implementation requires us to do use the updated jar files in our domain.

1. Copy the following jar files (can be found in AquaLogic Commerce Services WAR files) into the WebLogic domain lib directory i.e. %domain_home%/lib

```
stringtemplate-stringtemplate-2.3b6.jar
javax-saaj-api-2.1.1.jar
javax-saaj-impl-1.3.jar
javax-jsr181-api-2.1.1.jar
```

2. Edit the setDomainEnv.sh/setDomainEnv.cmd and append these jar files to the preclasspath so they are used instead of the libraries included inside WebLogic

```
PRE_CLASSPATH=%DOMAIN_HOME%\lib\stringtemplate-stringtemplate-2.3b6.jar;%DOMAIN_HOME%\lib\javax-saaj-api-2.1.1.jar;%DOMAIN_HOME%\lib\javax-saaj-impl-1.3.jar;%DOMAIN_HOME%\lib\javax-jsr181-api-2.1.1.jar
```

3. You will also need to remove the class load filtering defined in %application directory%/META-INF/weblogic-application.xml by deleting this file.

WebLogic Server 10 instructions

1. The **stringtemplate-stringtemplate-2.3b6.jar** should be added to the preclasspath so that the search server works correctly. See above on how this is done.

Web Services application on WebLogic

WebLogic includes a different version of a JSR_173 API class (Qname.class) than the Web Services application requires. To get around this, it must be deployed inside an exploded EAR so that application filtered classloading can be used. The following steps will allow you to deploy the Web Services application on WebLogic:

1. Create a **webservices.ear/** directory and put the unzipped webservices.war directory structure inside it.
2. Create a **META-INF/** directory in the **webservices.ear/** directory; inside it create a weblogic-application.xml file and an application.xml file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-application xmlns="http://www.bea.com/ns/weblogic/90">
<prefer-application-packages>
        <package-name>javax.jws.*</package-name>
        <package-name>javax.xml.soap.*</package-name>
        <package-name>com.sun.xml.*</package-name>
        <package-name>javax.xml.bind.*</package-name>
</prefer-application-packages>
</weblogic-application>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN"
"http://java.sun.com/dtd/application_1_3.dtd">

<application>

  <display-name>AquaLogic Commerce Web Services</display-name>

  <description>AquaLogic Commerce Web Services</description>

  <module>

        <web>

                <web-uri>webservices</web-uri>

                <context-root>webservices</context-root>

        </web>

  </module>

</application>
```

For AquaLogic Commerce Web Services add this block after the <resource-env-ref> section in the Web Services web.xml file:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

This is a work-around that will fix a double login problem in WebLogic 9.2. This work-around is not ideal since no CLIENT-CERT is actually configured, but this issue will be fixed in later versions of WebLogic or AquaLogic Commerce Web Services.

Finally, delete xerces-xercesimpl-2.8.0.jar and xerces-xmlapis-2.8.0.jar from the WEB-INF\lib directory of the Commerce Manager Server and, if applicable, the Web Services application.

Xerces jar conflicts

WebLogic Server ships with alternate versions of the xerces libraries but "Prefer WEB-INF classes" was selected for the Web Services application, so to avoid an IncompatibleClassChangeException you must remove the two xerces jar files from the CM Server and the Web Services web applications.

Troubleshooting

LinkageError: Class javax/xml/soap/MessageFactory violates loader constraints

This error shows up in the logs after configuration of web services and affects a built-in WebLogic application, but appears to have no effect on the correct functioning of the application.

Authorize.net payment processing

When trying to connect to the Authorize.net testing server with WebLogic, WebLogic gives an error similar to the following:

```
<Warning> <Security> <BEA-090504> <Certificate chain received from test.authorize.net
- 64.94.118.75 failed hostname verification check. Certificate contained \*.authorize.net
but check expected test.authorize.net>
```

The workaround for this is to turn off "hostname verification" in WebLogic. This is ONLY an issue with test transactions and does not occur when connecting to the Authorize.net production server.

StAX MBean exceptions

If you are getting javax.xml MBean or com.bea.xml exceptions on WebLogic 9.2 on startup, it may be that the domain is missing a copy of StAX jar libraries. Drop a copy of the latest StAX distribution (available from http://stax.codehaus.org/) in your domain/lib directory to resolve this.

WebLogic Authentication Plug-in

The WebLogic authentication plug-in allows the Storefront module to authenticate against WebLogic Server, and also allows Single Sign-On (SSO) with WebLogic Server and WebLogic Portal applications running within the same server instance. This section provides the steps for setting up the authentication plug-in.

Copy the plug-in jar file

The first step is to copy the WebLogic authentication plug-in jar file com.elasticpath.plugins.wls_authenticator-6.0.jar to the AquaLogic Commerce Services Storefront WEB-INF/lib folder, e.g. into C:\Warfiles\commerceServices\WEB-INF\lib.

Copy the JAAS login configuration file

Copy the file jaaslogin.config into your domain root folder, e.g. <BEA_HOME>\user_projects\domains\commerceServices

Set the location of the login configuration file

1. For the application to be able to find the JAAS login configuration file we need to add a startup option.
2. Navigate to the location of your domain. For our example, <BEA_HOME>\user_projects\domains\commerceServices
3. Open the 'bin' directory. For our example, this will be <BEA_HOME>\user_projects\domains\commerceServices\bin.
4. Open the 'StartWebLogic.cmd' file in a Text Editor. (Note: please make sure this is the StartWebLogic.cmd inside the bin directory)
5. Add this line right before "set CLASSPATH=%CLASSPATH%;%MEDREC_WEBLOGIC_CLASSPATH%":

```
set JAVA_OPTIONS=%JAVA_OPTIONS% -
Djava.security.auth.login.config==jaaslogin.config
```

> There does need to be two equal signs in the line above, since the second equal sign indicates that this login file should override any others.

6. Save the file and exit your text editor.

Change the user ID generation mode

To allow customers to log into WebLogic Server after they change their email address, the user ID mode needs to be changed with the following steps.

1. Open the WEB-INF\commerce-config.xml under your Storefront web application root folder and change the "userId.mode" value to "2".
   In this mode, the system will generate a unique user id for each customer. This user id will not be changed when customers log into the Storefront to
   change their email address.
2. Repeate last step for your Commerce Manager application.

Change user maintenance mode

By default, the Commerce Manager does not support the maintaintenance of a password when using WebLogic authentication. To change this, the user maintenance mode needs to be changed as follows.

1. Open the WEB-INF\commerce-config.xml under your Commerce Manager root folder.
2. Change the "user.maintain.mode" to "2".
   In this mode the create/delete customer and edit password functions will be disabled.

Acegi.xml

Ensure that your acegi.xml file uses the correct ports for http and https.

Single Sign-on Cookie

To ensure Single Sign-on works without problems with your WebLogic Server and WebLogic Portal applications, ensure that the weblogic.xml file for those applications contains the following settings:

```
<session-descriptor>
  <cookie-name>JSESSIONID</cookie-name>
  <cookie-path>/</cookie-path>
</session-descriptor>
```

WebLogic Portal Unified User Profile Plug-in

WebLogic Portal supports the concept of a Unified User Profile (UUP) which allows a Portal user to be associated with profile data from external systems. The UUP plug-in provides an EAR file and user profile property set that allows you to manage AquaLogic Commerce Services user properties from the WebLogic Portal Administration Console or within WebLogic Portal code. The details on how to manage user properties from an external system with the WebLogic Portal is described in the WebLogic Portal User Management Guide at http://edocs.bea.com/wlp/docs102/users/index.html.

Assumptions

This section assumes you already know how to set up a WebLogic Portal project. Refer to the Getting Started with WebLogic Portal documentation at http://edocs.bea.com/wlp/docs102/tutorials/index.html for a tutorial on setting up your Portal environment and creating a Portal.

Installing the plug-in

Perform the following steps in a new or existing Workshop for a WebLogic Platform Portal Ear Project:

1. If a Datasync project has not already been created in association with the EAR, create one (right click, New > Datasync Project) and associate it with the EAR.
2. Copy CommerceServices.usr to the userprofiles folder of your Datasync project
3. Enter Workshop's Project Explorer view (Window > Show View > Project Explorer) and expand Enterprise Applications.
4. Expand the node relating to your Portal EAR
5. Expand the [WebLogic] node
6. Right-click WebLogic J2EE Libraries and choose Add
7. In the Add WebLogic J2EE Library reference window, click Browse
8. In the Select WebLogic J2EE Library window, click Add
9. In the Add WebLogic J2EE Library window, click Browse
10. Browse to the plugin's alcs-uup-lib-6.0.ear and click Open.
11. Click OK in the remaining open windows.
12. Deploy and run your Portal EAR as usual.

You should now be able to access the WebLogic Portal UUP in the Portal Administration Console.

User creation, deletion, and sharing

The UUP plugin fully supports the Portal User Management feature for creating end users. This functionality in effect will also create and remove users from the ALCS database.

When creating users, the ALCS UUP plugin uses the default store code to register new user with a login store. This setting can be edited at the profile management console to change the user's default login store.

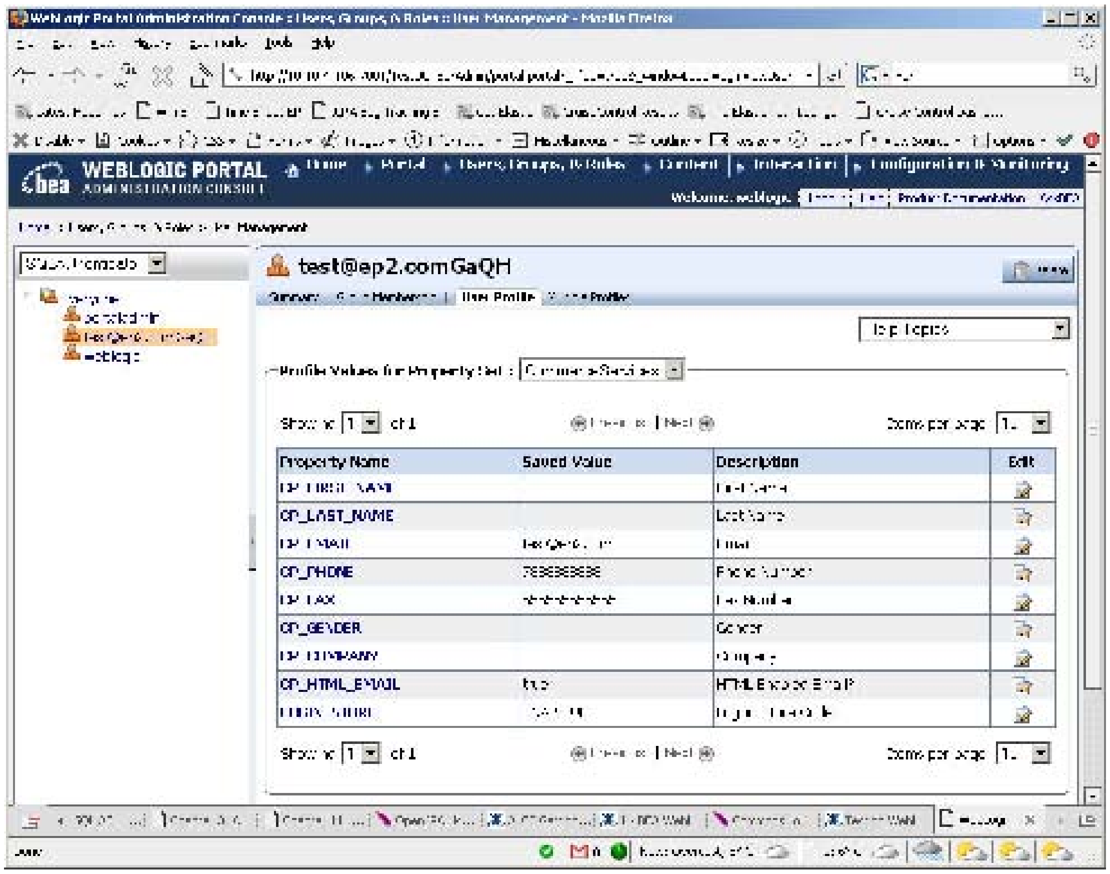
The default store code is specified in the UUP EAR within the file **WEB-INF/applicationContext.xml**, as follows:

```
<property name="defaultStoreCode"><value>SNAPITUP</value></property>
```

Please refer to the Commerce Manager User Guide for further information on enabling user sharing between stores.

Note: The UUP plugin creates users in the ALCS database only when a specific user profile is selected. This means that, at the time of creation of a new portal user, no ALCS user profile is created.

The ALCS user profile can be set up by choosing User Profile -> CommerceServices for a portal user (see screenshot below).



test@ep2.comGaQH

Profile Values for Property Set : CommerceServices

Property Name	Saved Value	Description	Edit
CP_FIRST_NAME		First Name	
CP_LAST_NAME		Last Name	
CP_EMAIL	[illegible]	Email	
CP_PHONE	[illegible]	Phone Number	
CP_FAX	[illegible]	Fax Number	
CP_GENDER		Gender	
CP_COMPANY		Company	
CP_HTML_EMAIL	true	HTML Enabled Email?	
[illegible]	[illegible]	[illegible]	

JNDI and JDBC Configuration

This section discusses how to configure your JNDI and JDBC settings to connect your application server to your database.

JNDI Configuration

The default JNDI name used for the AquaLogic Commerce Services data source is "*jdbc/epjndi*". This should not be changed. If you must use a different JNDI name, you must change this in all locations where it is listed in this document.

In addition, the JNDI name must be changed in the following places.

- <WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/web.xml
- <WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/web.xml
- <WL_HOME>/samples/commerce/commerceApp/commerceServicesConnect/WEB-INF/web.xml

JDBC Configuration

The configuration information for the different databases is listed in this section.

MySQL

Database Driver JAR file: mysql-connector-java-3.1.11-bin.jar
Database Driver: com.mysql.jdbc.Driver
Database Connection URL:
jdbc:mysql://*HOSTNAME:PORT/DBNAME*?autoReconnect=true&useUnicode=true&characterEncoding=UTF8
(Default Port is 3306)
Database Driver Download: http://dev.mysql.com/downloads/connector/j/3.1.html

Oracle

Database Driver JAR file: ojdbc14.jar
Database Driver: oracle.jdbc.driver.OracleDriver
Database Connection URL: jdbc:oracle:thin:@_HOSTNAME:PORT:SID
(Default port is 1521)
Database Driver Download:
http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc9201.html

Microsoft SQL Server (2000)

Database Driver JAR file: msutil.jar, msbase.jar, mssqlserver.jar (all 3 are required)
Database Driver: com.microsoft.jdbc.sqlserver.SQLServerDriver

Database Connection URL:
jdbc:microsoft:sqlserver://*HOSTNAME:PORT*;DatabaseName=*DBNAME*;selectMode=cu
rsor
(Default port is 1433)
Database Driver Download:
http://www.microsoft.com/downloads/details.aspx?FamilyID=07287b11-0502-461a-
b138-2aa54bfdc03a&DisplayLang=en

Configuring AquaLogic Commerce Services

This section explains how to configure the various AquaLogic Commerce Services applications.

Each sub-section explains the common configuration parameters. There are many other configuration parameters not described here which are further documented in the comments of the various configuration files listed. These other parameters can use their default values and should only be modified if needed.

Storefront

Configure the following files for your AquaLogic Commerce Services Storefront application:

- acegi.xml
- commerce-config.xml
- quartz.xml
- velocity.xml
- util-config.xml

acegi.xml

AquaLogic Commerce Services uses the Acegi security framework to perform user authentication and authorization.

The acegi.xml file for configuring acegi is located in the following application-specific paths. Since the acegi.xml file contains different settings for each application, you will need to configure each of them separately.

- Storefront – <WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/conf\spring/security/
- Commerce Manager – <WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/conf\spring/security/

- Web Services – <WL_HOME>/samples/commerce/commerceApp/commerceServicesConnect/WEB-INF/conf\spring/security/

Configuring the HTTPS port redirect

Specify your HTTP port number in the <entry> tag as described in the example below.

In acegi.xml, specify the HTTP port number in the key attribute of the entry tag and the HTTPS port number in the <value> tag as shown in the example below. This allows the application server switch to the HTTPS port for pages that require it. Not all URLs in an application are secure. For each application, we have specified a default list of URLs which are secure. You can modify these lists if desired as explained in the Acegi - security framework section of the AquaLogic Commerce Services Developer Guide.

In the example below, we have specified the standard port numbers of 80 for HTTP and 443 for HTTPS.

```
<!-- port # are specified in default.xml -->
    <bean id="portMapper" class="org.acegisecurity.util.PortMapperImpl">
   <property name="portMappings">
     <map>
            <entry key="80"><value>443</value></entry>
        </map>
     </property>
    </bean>
```

intelligent-browsing.xml

The AquaLogic Commerce Services Storefront uses intelligent-browsing.xml to generate the Filtered Navigation menu to display to users when they are browsing a category or executing a search.

Filtered Navigation appears in the storefront user interface as a set of links on the left side of the pages. Users can click links in the Filtered Navigation area to filter the list of products to find the product that they are looking for.

The guided-navigation.xml file is located in the following location:
<WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/

There are 3 types of filters defined in elements in the configuration file:

- attribute – defines the simple value filters for an attribute.
- attributeRange – defines the range values for an attribute.

- price – defines the price ranges.

Attribute Simple Value Filter

This defines the simple value filter for an attribute. It starts with the "attribute" tag as shown in the following example.

```
<attribute key="A00551" localized="false">
  <simple id="01" value="CCD" />
  <simple id="02" value="CMOS" />
  <simple id="03" value="Super HAD CCD" />
  <simple id="04" value="Live MOS" />
  <simple id="05" value="3CCD" />
  <simple id="06" value="LBCAST" />
</attribute>
```

- key="A00551" is the reference to the attribute key. Each attribute can only be defined once in the XML file.
- localized="false" means that the this attribute is not localized.
- <simple id="01" value="CCD" /> defines each available value. The "id" should be unique for this attribute since it is used to reference the filter. The "value" should be unique too.

If the attribute is localized, it will have the following differeneces in the XML.

- The "localized" property should be "true".
- For each simple value, it should have the "language" property defined.
- The "id" should be unique for this attribute since it is used to reference this filter, but the "value" can be the same for different languages.

The following is a localized attribute example.

```
<attribute key="A00556" localized="true">
  <simple id="01" value="TFT active matrix" language="en" />
  <simple id="02" value="Matrice active TFT" language="fr" />
  <simple id="03" value="LCD passive matrix" language="en" />
  <simple id="04" value="LCD à matrice passive" language="fr" />
  <simple id="05" value="None" language="en" />
  <simple id="06" value="Aucun(e)" language="fr" />
  <simple id="07" value="LCD" language="en" />
  <simple id="08" value="LCD" language="fr" />
</attribute>
```

Attribute Range Filter

This defines the range value for an attribute. Only attributes with type "Integer", "Decimal" or "Short-Text" can have ranges defined. It starts with the "attributeRange" tag. The range filter can have subranges defined.

```
<attributeRange key="A00140" localized="false">
 <range lower="" upper="1.9" id="_1.9">
  <display language="en">
   <value>1.9 in. &amp; Under</value>
   <seo>1.9in-and-under</seo>
  </display>
  <display language="fr">
   <value>1.9 in. et dessous</value>
   <seo>1.9in-et-dessous</seo>
  </display>
 </range>
 <range lower="2" upper="2.9" id="2_2.9">
  <display language="en">
   <value>2 to 2.9 in.</value>
   <seo>2-to-2.9in</seo>
  </display>
  <display language="fr">
   <value>2 et 2.9 in.</value>
   <seo>2-et-2.9in</seo>
  </display>
 </range>
 <range lower="3" upper="" id="3_">
  <display language="en">
   <value>3in. &amp; Up</value>
   <seo>3in-and-up</seo>
  </display>
  <display language="fr">
   <value>3in. et Plus</value>
   <seo>3in-et-plus</seo>
  </display>
 </range>
</attributeRange>
```

The attribute range filter also has the "key" property and "localized" property. Each "range" tag represents one range, which may have the "lower" value and "upper" value set. If the "lower" value is not set, then there is no minimum allowed value for the attribute filter. If the "upper" value is not set, then the there is no maximum allowed value for the attribute filter.

The "display" tag contains the information to display the filter on the guided navigation links.

Price Filter

This defines the range value for the price. It starts with the "Price" tag. The range filter can have subranges defined.

```
<price currency="USD" localized="false">
  <range lower="" upper="100" id="_100">
    <display language="en">
      <value>Under $100</value>
      <seo>under-$100</seo>
    </display>
    <display language="fr">
      <value>au-dessous de $100</value>
      <seo>au-dessousd-de-$100</seo>
    </display>
  </range>
  <range lower="100" upper="300" id="100_300">
    <display language="en">
      <value>$100 to $300</value>
      <seo>$100-to-$300</seo>
    </display>
    <display language="fr">
      <value>$100 et $300</value>
      <seo>$100-et-$300</seo>
    </display>
    <range lower="100" upper="200" id="100_200">
      <display language="en">
        <value>$100 to $200</value>
              <seo>$100-to-$200</seo>
      </display>
      <display language="fr">
```

```
        <value>$100 et $200</value>
            <seo>$100-et-$200</seo>
      </display>
    </range>
    <range lower="200" upper="300" id="200_300">
      <display language="en">
        <value>$200 to $300</value>
            <seo>$200-to-$300</seo>
      </display>
      <display language="fr">
        <value>$200 et $300</value>
            <seo>$200-et-$300</seo>
      </display>
    </range>
  </range>
  <range lower="300" upper="" id="300_">
    <display language="en">
      <value>More than $300</value>
      <seo>more-than-$300</seo>
    </display>
    <display language="fr">
      <value>au-dessous de $300</value>
      <seo>au-dessousd-de-$300</seo>
    </display>
  </range>
</price>
```

The price filter has the "currency" and "localized" property. Each "range" tag represents one price range, which may have the "lower" and "upper" values set. If the "lower" value is not set, then there is no minimum allowed value for the price filter. If the "upper" value is not set, then there is no maximum allowed value for the price filter.

The "display" tag contains the information to display the filter on the guided navigation links.

commerce-config.xml

AquaLogic Commerce Services's configuration settings are stored in the commerce-config.xml files located in the following application-specific paths.

Storefront:

<WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/

Commerce Manager:

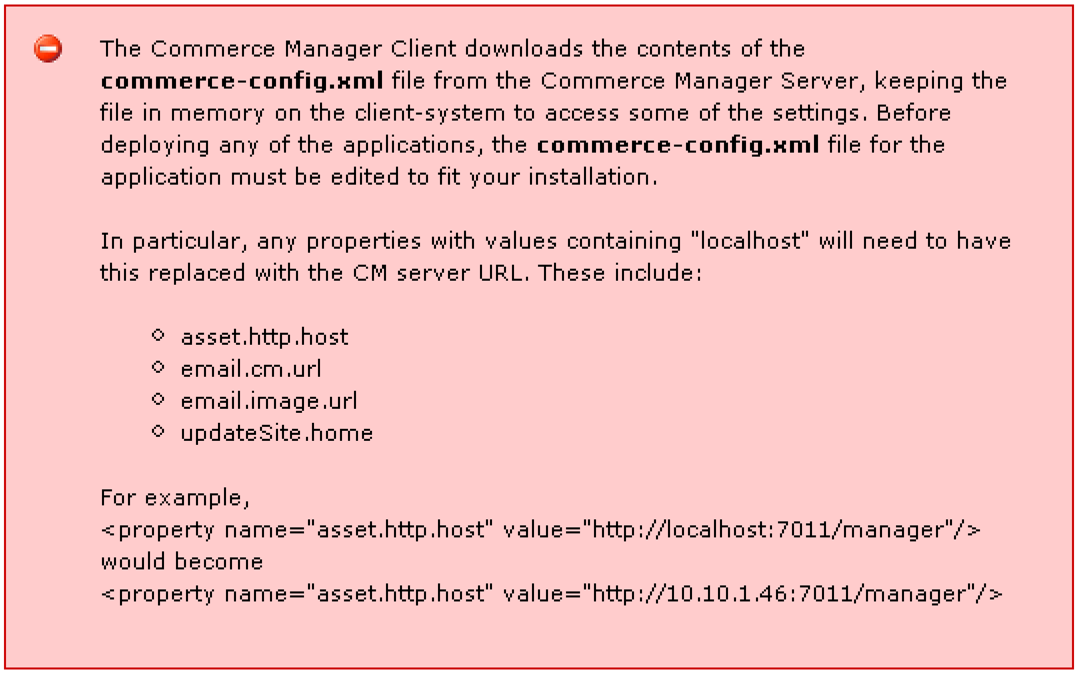
<WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/

Web Services:

<WL_HOME>/samples/commerce/commerceApp/commerceServicesConnect/WEB-INF/

Search Server:

<WL_HOME>/samples/commerce/commerceApp/commerceServicesSearch/WEB-INF/

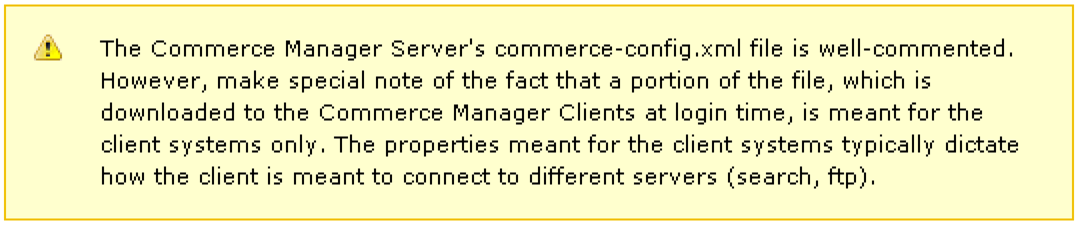


The Commerce Manager Client downloads the contents of the **commerce-config.xml** file from the Commerce Manager Server, keeping the file in memory on the client-system to access some of the settings. Before deploying any of the applications, the **commerce-config.xml** file for the application must be edited to fit your installation.

In particular, any properties with values containing "localhost" will need to have this replaced with the CM server URL. These include:

- asset.http.host
- email.cm.url
- email.image.url
- updateSite.home

For example,
<property name="asset.http.host" value="http://localhost:7011/manager"/>
would become
<property name="asset.http.host" value="http://10.10.1.46:7011/manager"/>



The Commerce Manager Server's commerce-config.xml file is well-commented. However, make special note of the fact that a portion of the file, which is downloaded to the Commerce Manager Clients at login time, is meant for the client systems only. The properties meant for the client systems typically dictate how the client is meant to connect to different servers (search, ftp).

The following are the names and descriptions of the key elements in the configuration file.

- commerceConfig - the main element containing all the configuration settings for an AquaLogic Commerce Services implementation.
    - web - the element containing all web-related settings.

        - web.sf.context.url - the context part of the URL for the Storefront application. For example, "alcssf". Set this to an empty string if the application is in the root context.
        - catalog.topSeller.count - the number of top selling products to display on the category pages
        - catalog.view.pagination - the number of items to display per page on the catalog browsing and search pages
    - customer - the element containing customer settings
        - userid.mode - determines whether the customer's email address will be used as userid or a unique userid will be generated based on the email address
    - asset - the element containing the paths to all assets
        - catalog.asset.path - the root path where catalog assets are stored
        - image.asset.subfolder - the subdirectory under the root path where images are stored
        - file.asset.subfolder - the subdirectory under the root path where non-image files are stored
        - digitalgoods.asset.subfolder - the subdirectory under the root path where digital goods (such as electronic books) are stored
    - mail - the element containing all mail-related settings.
        - mail.host - domain or IP address of your SMTP server. This is used to send email confirmations.
        - mail.host.port - the port of the SMTP server.
        - email.from - the "From" email address for all emails generated by AquaLogic Commerce Services, e.g. customerservice@elasticpath.com.
        - email.from.name - the "From" personal name for all emails generated by AquaLogic Commerce Services, e.g. AquaLogic Commerce Services.
        - email.store.url - URL for images etc. in emails generated by the Storefront
        - email.cm.url - URL for images etc. in emails generated by the Commerce Manager
    - units - the element containing the settings for units of measurement
    - units.length - the unit of length to display. For example, "CM" for centimeters.
    - units.weight - the units of weight to display. For example, "KG" for kilograms.
    - payment - the element containing information about configured payment gateways. Configure your available payment methods here.

- seo - the element containing search engine optimization (SEO) settings, which generates search engine friendly URLs for products and categories
  - seo.enable - enables/disables SEO URL generation
- attributeFilter - the element containing settings for guided navigation, which allows customers to narrow down their browsing/search results to find a product that they are looking for
  - attributeFilter.enable - enables/disables guided navigation
- dynamicimagesizing - the element for setting image display options
  - dynamicimagesizing.jpegquality - the quality of the displayed jpeg images
  - dynamicimagesizing.noimage - the name of the image to display if the image to be displayed is not found
- search - the element containing catalog search settings
  - maxReturnNumber - the maximum number of products to return in a search result
  - minimumSimilarity - used in searches that do not require an exact match (fuzzy searches). This is the fraction of similarity required between the search term and a matching term
  - prefixLength - used in searches that do not require an exact match (fuzzy searches). This is the number of characters at the beginning of the term that must match exactly between the search term and matching term
  - minimumResultsThreshold - the number of results under which alternate queries (such as spelling suggestions) will be suggested to the user
  - maximumSuggestionsPerWord - the maximum number of suggestions generated for each word in the query
  - accuracy - the fraction of similarity required between suggested words and original words in the query
- onepage - the element containing settings for One Page checkout
  - onepage.enable - the boolean true/false value specifying if One Page checkout is enabled or not
- powerreviews - the element containing settings for PowerReviews product reviews
  - powerreviews.enable - the boolean true/false value specifying if PowerReviews is enabled or not
  - powerreviews.merchantid - the merchant account id provided by PowerReviews
- security - the element containing encryption information

        - encryption.key - the encryption key for encrypting customer credit card numbers stored in the TORDERPAYMENT table. You must configure this before order checkout will work.
    - productcache - the element containing product cache settings
        - productcache.preload - the boolean true/false value specifying if all products will be pre-loaded into the cache upon startup or not

Customer User Id Generation

AquaLogic Commerce Services by default uses a customer's email address as the user id, but you can change the user id generation mode in commerce-config.xml as shown below.

```
<property name="userid.mode" value="1" />
```

The possible values for the user id generation mode are:

1 - Use user's email as the user id, this is the default value. If you use JAAS, you can't use this mode.

2 - Generates a unique permanent user id by appending a random four digit suffix to the email address and uses this as the user id. The user id is created when the customer first creates an account. Later, when the customer changes the email address, the user ID will not be changed. For example, if a customer email address is a@a.com, the user id would be a@a.comXXXX, where XXXX is a random generated string.
3 - Independent email and user id. This mode is not yet supported.

Customer Maintenance Mode

This setting indicates whether to add, delete and maintain user passwords through the Commmerce Manager. By default the customer password is maintained in the AquaLogic Commerce Services database. If customers are authenticated through JAAS, (e.g. authenticated in WebLogic Server) the customer maintenance mode must be set to 2. This setting will only affect the Commerce Manager, not the Storefront.

```
<property name="user.maintain.mode" value="1" />
```

The possible values are:

1 - Maintain user passwords in the local DB, this is the default value.

2 - Maintain user passwords through JAAS, the Commerce Manager will disable the create/delete and change Password functions in this mode.

Payment Configuration

The payment gateway used by the Storefront is configured in the <payment> block within the commerce-config.xml file as follows.

- The checkout transaction behavior can be set to one of the following two values:

    - authorization - When an order is placed, the payment will be pre-authorized only. The payment will be captured later when the shipment for the goods is released. This mode is typically used whenever a store sells shippable goods.
    - sale - When an order is placed, the payment for the order will be captured immediately. This mode is typically used when only digital goods can be purchased in the store.
- The <gateway> block specifies the fully-qualified class name of the payment processor to be used, which must implement PaymentGateway.
    - The names of supported card types are also specified in this block. Note that the supported card types are for display only, since the payment processor determines the card type from the card number.
    - The ValidateCvv2 element determines whether the card security code should be requested from the user and sent to the payment processor.
- Only ONE "credit card processing" payment gateway is supported at a time. Changing to a different payment gateway requires re-configuring the payment gateway settings in the commerce-config.xml file.
- PayPal Express Checkout can also be configured alongside the chosen "credit card processing" payment gateway in the <payment> block.
  Each gateway has its own gateway-specific properties.

CyberSource Credit Card Sample Configuration

```
<gateway name="CyberSource"
class="com.elasticpath.domain.payment.impl.CyberSourcePaymentGatewayImpl">
    <property name="merchantID" value="YOUR_MERCHANT_ID"/>
    <property name="keysDirectory" value="RELATIVE/PATH/TO/CERT/DIRECTORY (from
WEB-INF)" />
    <property name="targetAPIVersion" value="1.19"/>
    <property name="sendToProduction" value="false"/>
    <property name="logMaximumSize" value="10"/>
    <property name="enableLog" value="false"/>
    <property name="logDirectory" value="WEB-INF/log"/>
</gateway>
```

Paypal Payflow Pro (formerly Verisign) Sample Configuration

```
<gateway name="PayflowPro"
class="com.elasticpath.domain.payment.impl.PayflowProPaymentGatewayImpl">
   <property name="user" value="YOUR_USERNAME"/>
   <property name="password" value="YOUR_PASSWORD"/>
   <property name="vendor" value="YOUR_VENDOR"/>
   <property name="partner" value="Verisign"/>
```

```
    <property name="certificateLocation" value="RELATIVE/PATH/TO/CERT/DIRECTORY
(from WEB-INF)"/>

    <property name="hostAddress" value="test-payflow.verisign.com"/>

    <property name="hostPort" value="443"/> <!--optional - defaults to 443-->

    <property name="proxyAddress" value=""/> <!--optional-->

    <property name="proxyPort" value=""/> <!--optional-->

    <property name="proxyLogon" value=""/> <!--optional-->

    <property name="proxyPassword" value=""/> <!--optional-->

</gateway>
```

Authorize.Net Sample Configuration

```
<gateway name="AuthorizeNet"
class="com.elasticpath.domain.payment.impl.AuthorizeNetPaymentGatewayImpl">

    <property name="authorizeNetURL"
value="https://certification.authorize.net/gateway/transact.dll"/>

    <property name="testMode" value="true"/>

    <property name="loginID" value="XXXX"/>

    <property name="transKey" value="XXXX"/>

    <property name="version" value="3.1"/>

    <property name="delimChar" value="|"/> <!--optional-->

    <property name="encapChar" value=""/> <!--optional-->

</gateway>
```

Fuzzy Search in Lucene

The search mechanism in AquaLogic Commerce Services takes advantage of the Fuzzy Search feature in Lucene. Fuzzy Search provides a way to search for terms similar to a specified term (for example with misspelled words).

The Fuzzy Search can be configured with two settings in the commerce-config.xml

1. minimumSimilarity - Value between 0 and 1 to set the required similarity between the query term and the matching terms. For example, for a minimumSimilarity of 0.5 a term of the same length as the query term is considered similar to the query term if the edit distance between both terms is less than length(term)*0.5 where edit distance is a measure of similarity between two strings and distance is measured as the number of character deletions, insertions, or substitutions required to transform one string to the other string.

2. prefixLength - length of common non-fuzzy prefix

Spelling Suggestions

Spelling suggestions can be generated for Storefront keyword queries. The Lucene Spellchecker is used to find words similar to terms in the original keyword query. This is configured in the following elements.

- minimumResultsThreshold - If the number of results does not exceed this setting the search will attempt to suggest possible alternate queries
- maximumSuggestionsPerWord - The maximum amount of suggestions that will be generated for each word in the query
- accuracy - The degree of similarity that suggested words will have to the original keywords

Encryption Key

For security reasons, this key field is not pre-set to any particular value out-of-the-box. You must make up your own site-specific value, and set this field. Otherwise, the application will NOT work with the empty default encryption key. Any string at least 24 characters may be used for the key. This must be entered in all AquaLogic Commerce Services application commerce-config.xml configuration files (Storefront, Commerce Manager, and Web Services).

> The same key must be used in all places in order for the credit card numbers that have been encrypted in the Storefront to be correctly decrypted and masked in the other applications.

One Page Checkout

One Page checkout is included as part of the AquaLogic Commerce Services install. It is enabled by default by setting "onepage.enable" to "true". You can always revert back to using the standard checkout process by changing this value to "false".

```
<onepage>
  <property name="onepage.enable" value="false"/>
</onepage>
```

> One Page checkout does not currently support purchase/redemption of gift certificates or payment via PayPal Express. If you wish to enable these features in the storefront, you will need to enable the standard checkout process as described above.

PowerReviews Product Reviews (optional)

You must have setup an account with PowerReviews to use this functionality. Once setup correctly, PowerReviews product reviews can be enabled by setting "powerreviews.enable" to "true". You can disable PowerReviews product reviews by setting this value to "false" and no product review information will display on the Storefront.

AquaLogic Commerce Services trial edition comes setup with a demo PowerReviews merchant account with some sample reviews, but any reviews created with this demo merchant id will not display on your Storefront. Once you have setup an account with PowerReviews, replace the value of "powerreviews.merchantid" with the merchant id assigned to you by PowerReviews.

```
<powerreviews>
  <property name="powerreviews.enable" value="true"/>
  <property name="powerreviews.merchantid" value="7609"/>
</powerreviews>
```

quartz.xml

Quartz is used in AquaLogic Commerce Services to execute scheduled jobs (for example, building the Lucene index). A quartz.xml file is used to configure quarz jobs for the different AquaLogic Commerce Services applications. The quartz.xml file is located in the following application-specific paths.

- **Storefront** – <WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/conf/spring/scheduling/
- **Commerce Manager** – <WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/conf/spring/scheduling/
- **Web Services** – <WL_HOME>/samples/commerce/commerceApp/commerceServicesConnect/WEB-INF/conf/spring/scheduling/

Quartz files have three basic types of blocks of XML:

- **Factory block** – The definition of the Scheduler Factory, which is used to populate a Scheduler class with the list of Triggers that it will be responsible for executing.
- **Job block** – The definition of the job that is being scheduled. This defines the method that will be called by the Trigger and references the class in which the method lives.
- **Trigger block** – The definition of the trigger that will be used to trigger the job. This will usually be either a SimpleTriggerBean (which runs a job every x milliseconds) or a CronTriggerBean (which will run a job at a specified time).

There is usually one Factory definition in a quartz.xml file, and then each job has both a Job definition and a Trigger definition.

Extended information about Quartz is available from the Quartz website at http://www.opensymphony.com/quartz/

Quartz Cron Configuration

The time/trigger to execute the scheduled job can be set with the cronExpression property. The cron expression contains six required components and one optional component. A cron expression is written on a single line and each component is separated from the next by space. Only the last, or rightmost, component is optional. The table below describes the cron components in detail.

Components of a Cron Expression:

Position	Meaning	Allowed Special Characters
1	Seconds (0-59)	, - * /
2	Minutes (0-59)	, - * /
3	Hours (0-23)	, - * /
4	Day of month (1-31)	, - * / ? L C
5	Month	(either JAN-DEC or 1-12) , - * /
6	Day of week (either SUN-SAT or 1-7)	, - * / ? L C #
7	Year (optional, 1970-2099), when empty, full range is assumed	, - * /

Each component accepts the typical range of values that you would expect, such as 0-59 for seconds and minutes and 1-31 for day of the month. For the month and day of the week components, you can use numbers, such as 1-7 for day of the week, or text such as SUN-SAT.

Each field also accepts a given set of special symbols, so placing a * in the hours component means every hour, and using an expression such as 6L in the day-of-the-week component means last Friday of the month. The table below describes cron wildcards and special symbols in detail.

Cron Expression Wildcards and Special Symbols:

Special Character	Description
*	Any value. This special character can be used in any field to indicate that the value should not be checked. Therefore, our example cron expression will be fired on any day of the month, any month, and any day of the week between 1970 and 2099.
?	No specific value. This special character is usually used with other specific values to indicate that a value must be present but will not be checked.
-	Range. For example 10-12 in the Hours field means hours 10, 11, and 12.
,	List separator. Allows you to specify a list of values, such as MON, TUE, WED in the Day of week field.
/	Increments. This character specifies increments of a value. For example 0/1 in the Minute field in our example means every 1-minute increment of the minute field, starting from 0.
L	L is an abbreviation for Last. The meaning is a bit different in Day of month and Day of week. When used in the Day of month field, it means the last day of the month (31st of March, 28th or 29th of February, and so on). When used in Day of week, it has the same value as 7---Saturday. The L special character is most useful when you use it with a specific Day of week value. For example, 6L in the Day of week field means the last Friday of each month.
#	This value is allowed only for the Day of week field and it specifies the nth day in a month. For example 1#2 means the first Monday of each month.
C	The Calendar value is allowed for the Day of month and Day of week fields. The values of days are calculated against a specified calendar. Specifying 20C in the Day of month field fires the trigger on the first day included in the calendar on or after the 20th. Specifying 6C in the Day of week field is interpreted as the first day included in the calendar on or after Friday.

Storefront Quartz Jobs

The demo storefront has one configured scheduled job.

SearchConfigUpdateJob

This job updates the search configuration settings in the storefront runtime memory from the search-config.xml file, so that changes to the search-config.xml file are reflected in the running storefront without requiring you to restart the storefront application. The job runs every 5 seconds in the demo store.

search-config.xml

The search-config.xml file is used to configure searches. It is located in the following location:

<WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF

The storefront need only know of 2 types of searches, category and product searches. Each type of search has the following common parameters:

- maxReturnNumber - the maximum number of products to return in a search result
- minimumSimilarity - used in searches that do not require an exact match (fuzzy searches). This is the fraction of similarity required between the search term and a matching term
- prefixLength - used in searches that do not require an exact match (fuzzy searches). This is the number of characters at the beginning of the term that must match exactly between the search term and matching term
- minimumResultsThreshold - the number of results under which alternate queries (such as spelling suggestions) will be suggested to the user
- maximumSuggestionsPerWord - the maximum number of suggestions generated for each word in the query
- accuracy - the fraction of similarity required between suggested words and original words in the query
- boosts - the boosts of particular field values in the index. Fields should only specified once. Locale fields have fallback so that french fields will fallback to the default field (in case french fields don't have boosts)

Product Searches

In the Storefront, searches also search through product attributes. You can specify attribute keys which should be excluded in searches. Having nothing excluded will result in all attributes being searched upon.

⚠ Setting the maxReturnNumber to a value other than 0 will result in browsing conforming to that constraint as well, i.e. setting a value of 500 will result in browses only showing the first 500 products.

velocity.xml

Velocity is used for UI html rendering in AquaLogic Commerce Services. The velocity.xml file for configuring Velocity is located in the following application-specific paths.

- Storefront – <WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/conf/spring/views/velocity/

- Commerce Manager – <WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/conf/spring/views/velocity/

Setting Velocity properties for a production environment

Changes made to velocity.xml should be applied to both the Storefront and Commerce Manager deployments.

Ensure that the cacheSeconds property is set to -1. When set to a positive value this property allows you to change and test changes to message source resource files without having to restart your application or servlet container. In production it is recommended that this property is disabled and set to -1.

```
<bean id="globalMessageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
  <property name="basenames">
    <list>
      <value>/WEB-INF/templates/velocity/globals</value>
    </list>
  </property>
  <!-- Set cacheSeconds to -1 in Production Environment -->
  <property name="cacheSeconds"><value>-1</value></property>
</bean>
```

The following properties in util-config.xml (Storefront and Commerce Manager) should also be set for optimal performance:

- file.resource.loader.cache - "true"
- velocimacro.library.autoreload - "false"
- velocimacro.messages.on - "false"

See the util-config.xml section for more details on these configuration options.

util-config.xml

This configuration file contains the Spring configuration for utility classes. It is located in the following application-specific paths.

- Storefront – <WL_HOME>/samples/commerce/commerceApp/commerceServices/WEB-INF/conf/spring/commons/
- Commerce Manager – <WL_HOME>/samples/commerce/commerceApp/commerceServicesManager/WEB-INF/conf/spring/commons/

Within the velocityProperties bean definition, the following properties can be configured.

- file.resource.loader.cache – Set this value to "true" for production to cache templates in Velocity's loader. Set this value to "false" for development so that changes to templates will take effect immediately.
- velocimacro.library.autoreload – This parameter enables/disables automatic reloading of Velocity macros. Set this value to "false" for production. For development, setting this value to "true" will allow changes to Velocity macros to take effect without restarting the server.
- velocimacro.messages.on – Set this value to "false" for production for improved performance. When developing Velocity templates, this value can be set to "true" to see Velocity logging and debugging information.
- runtime.log.logsystem.class – Set this value to org.apache.velocity.runtime.log.NullLogSystem as shown below for production.

```
<entry
key="runtime.log.logsystem.class"><value>org.apache.velocity.runtime.log.NullLog
System</value></entry>
```

Commerce Manager (CM)

Follow the Storefront instructions for configuring acegi.xml in your Commerce Manager application. Then configure the following files in your Commerce Manager application:

- commerce-config.xml
- quartz.xml
- velocity.xml
- util-config.xml

commerce-config.xml

See the Storefront commerce-config.xml section for the details on property configurations that are common to the Storefront, Commerce Manager, and Web Services applications.

The following are the names and descriptions of the key Commerce Manager elements the commerce-config.xml configuration file that differ from the Storefront commerce-config.xml.

- commerceConfig - the main element containing all the configuration settings for an AquaLogic Commerce Services implementation.
    - web - the element containing all web-related settings.

        - web.cm.context.url - the context part of the URL for the Commerce Manager application. For example, "alcscm". Set this to an empty string if the application is in the root context.
    - search - the element containing catalog search settings
        - maxReturnNumber - the maximum number of records returned in search results, such as customers and orders.
    - shipmentcarrier - the element containing the shipment carrier settings
        - carrier.all - the list of all shipment carriers, separated by commas
    - productrecommendations - the element containing product recommendation settings
        - productrecommendations.numorderhistorymonths - the number of months of previous order history that is used in product recommendation calcuations
        - productrecommendations.maxrecommendations - the maximum number of recommendations that will be computed for each product

> Setting either value to -1 prevents recalculation of the product recommendations. However, existing recommendations will NOT be removed as a result of setting the value to -1.

quartz.xml

The quartz.xml file is used to configure regularly scheduled jobs. See the Storefront quartz.xml section for a description of the basic structure of a quartz.xml file.

Commerce Manager Quartz Jobs

The Commerce Manager has three configured jobs:

- productRecommendationJob – recomputes product recommendations
- topSellerJob – determines the latest top selling products
- releaseShipmentTrigger – releases shipments for packing

Search Server Quartz Jobs

- The index build process will pick up all added, modified, and deleted objects since the last build and update the index accordingly.

> Since orders cannot be deleted from the system, the index build process won't pick up deleted orders.

- Commerce Manager index build scheduling should normally have a shorter lag time than the Storefront to ensure the index is as up-to-date as possible. The default setting is to start the first build 5 seconds after application server started and then build every 5 seconds.
- Indexes are created under the following subcategories of the search web application's WEB-INF/solrHome directory:
    - categoryIndex
    - productIndex
    - customerIndex
    - orderIndex

The Search Server has six configured jobs, one for each type of index the search server is indexing:

- customerIndexBuildJob - build up the customer index
- orderIndexBuildJob - builds up the order index
- orderReturnIndexBuildJob - builds up the order return index
- productIndexBuildJob - builds up the product index
- categoryIndexBuildJob - builds up the category index
- promotionIndexBuildJob - builds up the promotion index
- catalogPromoMonitorJob - checks for changes to catalog promotion rules and notifies the search server if any are found.
- rulebaseCompileJob - re-compiles the rule base, and stores it in the database

Computing Product Recommendations

A trigger is declared for computing product recommendation data. A SimpleTriggerBean is typically used to compute recommendations every 30 seconds for demonstration purposes. For production, a CronTriggerBean should be used to compute recommendations with a longer interval such as one day.

Computing the Top Sellers

A trigger is declared for computing the top selling product statistics. Like the product recommendation job, A SimpleTriggerBean is typically used to compute recommendations every 30 seconds for demonstration purposes. For production, a CronTriggerBean should be used to compute recommendations with a longer interval such as one day.

Releasing Shipments for Packing

Many companies find that there should be a "pick delay" between the time a person places an order and the time that the order becomes released to the warehouse for packing, so that people have time to change their orders before the orders are scheduled for shipping. The pick delay is configurable by Warehouse in the Commerce Manager Client. A CronTriggerBean for the ReleaseShipmentTrigger is used to release shipments for packing at a fixed time interval. Every time this job is run the system will find all shipments that were created at least X number of hours ago, where X is the pick delay, and will change their status to release them for picking if inventory is available.

search-config.xml

See the Storefront velocity.xml section for the properties that are common to the Storefront and Commerce Manager.

The only differences between this config file and the Storefront config file is that attributes are not searched for in the case of product searches (attribute exclusion will be ignored) and that the search types are specific to the Commerce Manager.

velocity.xml

See the Storefront velocity.xml section for the details on configuring velocity template properties for both the Storefront and Commerce Manager.

util-config.xml

See the Storefront util-config.xml section for the configuration options for both the Storefront and Commerce Manager.

Web Services

Follow the instructions in the following sections for configuring your Web Services application.

- Storefront acegi.xml – Configuring https security and adding new users / roles / permissions for web services.
- Commerce Manager commerce-config.xml – Configuring max number of results for order and customer search.
- Commerce Manager quartz.xml – Configuring the Lucene index for order and customer searches.

See the Commerce Manager manual for instructions on how to create users that can authenticate with Web Service.

One Page Checkout

One Page checkout is included and turned on by default in AquaLogic Commerce Services. See the commerce-config.xml section for the One Page checkout configuration options.

PowerReviews Product Reviews (optional)

The basic steps required to setup PowerReviews product reviews are as follows:

1. Setup a merchant account with PowerReviews (see the PowerReviews website at http://www.powerreviews.com/ for details)
2. Setup a new login on your FTP server for PowerReviews to send your scheduled ZIP file with review data and the latest JavaScript code
3. Create a script to unzip the contents of the PowerReviews ZIP file from your FTP server to the <WL_HOME>/samples/commerce/commerceApp/commerceServices\template-resources/power-reviews/ directory on your AquaLogic Commerce Services Storefront server (overwriting the existing directory)
4. Configure PowerReviews settings in AquaLogic Commerce Services (see the Storefront commerce-config.xml section for details)

Consult your PowerReviews documentation for more details on how to setup and customize PowerReviews.

Configuring for Optimal Performance

This section lists tips and settings for configuring AquaLogic Commerce Services for optimal performance.

The first sub-section outlines general performance tuning considerations that affect both the Storefront and Commerce Manager. The following sub-sections describe specific optimizations for the Storefront and Commerce Manger. The Advanced optimization section covers additional techniques for troubleshooting performance problems.

This document only covers performance topics from a configuration and deployment perspective.

Key configuration files and settings

Refer to the documentation of the following configuration files to ensure you have configured AquaLogic Commerce Services for a production environment. Remember to check these files in both your Storefront and your Commerce Manager deployments.

- **velocity.xml** - It is very important that you have configured Velocity for production according to these instructions.
- **util-config.xml** - It is very important that you have configured Velocity for production according to these instructions.
- **quartz.xml** - Check this file to ensure that you are not running batch jobs during peak times or during performance tests.

Performance tip
Check that you have configured Velocity for production by following the instructions in velocity.xml and especially util-config.xml.

General tips

Run your JVM in server mode

The JVM server mode will take longer to start but performs better once it is running. Use server mode for production environments and client mode for development only.

Specify the JVM heap size

By default, a JVM may only be able to use 64M or 128M memory.
You should specify the heap size of the application server JVM to make more memory accessible to it.

- -Xms specifies the initial Java heap size
- -Xmx specifies the maximum Java heap size

```
Sample setting:
-Xmx4096m -Xms4096m
```

Linux Memory Limitation

In 32bit Linux, the JVM may only be able to use up to 2GB memory.

Use a different garbage collector

In JVM version 1.5, several types of garbage collectors are provided. On a 2-CPU server, our test results show that the concurrent low pause collector gives a little better performance under stress.

Application Server

- Shorten the session timeout
  If you define a long session timeout, data stored in the session have a higher chance of being pushed up into slower segments of the JVM memory where they are expensive to garbage collect. Shorter timeouts will also increase the amount of memory available.

- Consider disabling session replication
  Session replication makes it possible to retain customer sessions when an application server is down. However, this will slow down application servers to some extent. If it's not critical to retain customer sessions, disabling replication can result in better performance.

Reverse Proxy Server

Reverse proxy servers are often used to cache static content to reduce load on application servers. AquaLogic Commerce Services implements an image renderer which can render an image in any desired size. This simplifies image management but also incurs CPU overhead each time an image is rendered. However there is a little cpu overhead for the application server to use this image renderer again and again on the same image. We recommend enabling caching for the rendered images on a reverse proxy server.

Infrastructure

Ensure that enough network bandwidth is provided between the following servers.

- Internet and the reverse proxy server
- Reverse proxy server and the application servers
- Application servers and the database server

Storefront

Periodically archive and delete old shopping carts

Shopping carts never expire in the database. This is good in the sense that you can create reports which track trends over a wider range of time, but the drawback is that, over time, **the database will grow large**. It is therefore important to periodically archive and delete old shopping carts for optimal database and storefront performance.

Advanced optimization

Monitor JVM memory usage in JVM 5.0

You can use the following tools that ship with JVM 1.5 to monitor memory usage.

```
jstat -gc JVM_PROCESS_ID MONITORING_INTERVAL
e.g. jstat -gc 1234 5000
```

How to get a thread trace

A thread trace will help you see thread status and find thread blockers.
You can use the following command in Linux to get a thread trace.

```
kill -3 JVM_PROCESS_PID
```

Integrating AquaLogic Commerce Services and WebLogic Portal

With the AquaLogic Commerce Services 6.0 release, several integration components have been developed to give you more power on creating customized applications.

The following components are available after AquaLogic Commerce Services 6.0 installation:

- WorkSpace Studio Facets – provides easier configuration and deployment of the integration modules;
- Content Management Integration – provides access to the product catalog through the WebLogic Portal Content Management framework;
- Sample Portlets – provides out-of-the-box portlets for WebLogic Portal to be integrated in your portal to show Shopping Cart items, Order History and Product Suggestions

For more information about how to deploy AquaLogic Commerce Serviecs in a WebLogic Portal domain, refer to the *AquaLogic Commerce Services Getting Started Guide*

WorkSpace Studio Facets

The WorkSpace Studio Facets are a set of plugins that can be installed in your WorkSpace Studio installation to give you quicker access to all AquaLogic Commerce Services integration modules, by configuring and copying the necessary files to your enterprise or web applications.

How to install

The WorkSpace Studio plugin is installed at the following location after a successful installation of AquaLogic Commerce Services:

```
BEA_HOME/commerce_6.0/eclipse/plugins
```

Copy all the content within the above location to:

```
WL_HOME/portal/eclipse/plugins
```

If your WorkSpace Studio is currently running, make sure to close it and reopen to make the new plugin available for usage.

After the above procedure is completed, you should now see all the facets available on your WorkSpace Studio installation. In the next topics on this guide, we will review how to localize all new available Facets to increase your productivity.

How to uninstall

Note: Before executing the following procedure, make sure you close WorkSpace Studio if it is currently running.

If you want to remove the facets from your WorkSpace Studio installation, delete the above copied content from the following location:

```
WL_HOME/portal/eclipse/plugins
```

Available Facets

The following facets will be available in your WorkSpace Studio after installation:

- **ALCS Core:** includes all AquaLogic Commerce Services core components required for integration
- **ALCS Storefront:** adds to your Enterprise Application the sample Storefront Web Application
- **ALCS WebServices Server:** adds to your Enterprise Application the WebServices Server that exposes many services through its SOAP and REST interfaces
- **ALCS WebServices Client:** adds to your Web Application the client library that binds to the WebServices easing your usage through POJO classes and Singleton services
- **ALCS Content Management Integration:** adds the required libraries and dependencies to your Enterprise Application. It also configures the required XML files for easier deployment

WebLogic Portal Content Management Integration (SPI)

The WebLogic Portal Content Management integration has been designed to give you the same experience and functionality as you would have using the Out-of-the-box Repository Services from WebLogic Portal.

Using this integration module, you will be able to retrieve category and product information using all functionality already available for you on WebLogic Portal: JSP Tags, Content Management API, campaigns and placeholders to personalize the presentation of content and catalog data.

The Content Management integration with AquaLogic Commerce Services is read-only, meaning that you will not able to change product information through the Portal Administration Console, JSP Tags, and API.

To update either category or product information, you should use the Commerce Manager application, installed with your AquaLogic Commerce Services installation. For more information, please refer to the Commerce Manager User Guide.

How to Install in Your Project

In order to install the Content Management integration, you need to have the WebLogic Portal Enterprise Application created prior to adding the Facet.

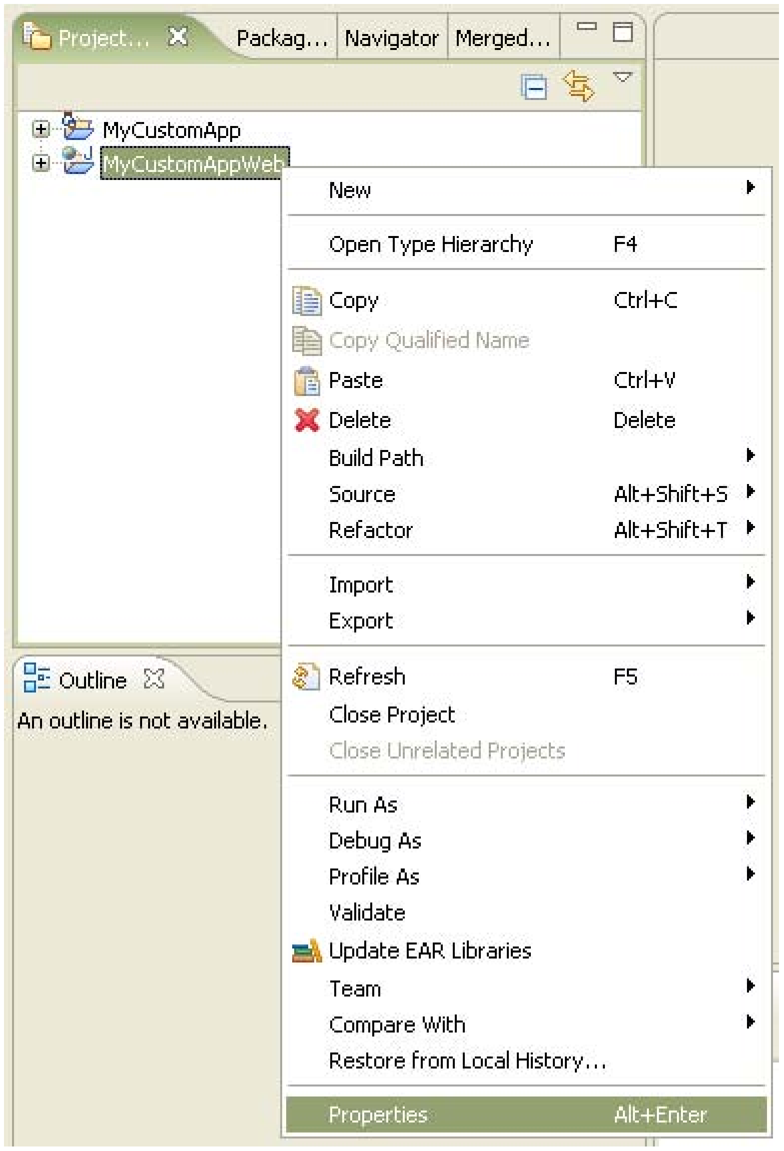
If you have not yet created your WebLogic Portal Enterprise Application, please refer to the Getting Started with WebLogic Portal documentation at http://edocs.bea.com/wlp/docs100/tutorials/index.html.

Installing by using WorkSpace Studio Facets

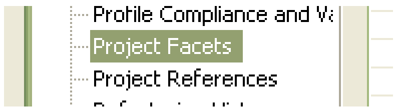
This is the recommended way to add Content Management integration to your application, as it will automatically add all required libraries and make the proper configurations on your WebLogic Portal Enterprise Application.

To add the Content Management integration Facet to your WebLogic Portal Enterprise Application, execute the following procedures:

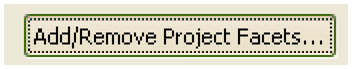
1. Right-Click on your WebLogic Portal Web Application and select Properties



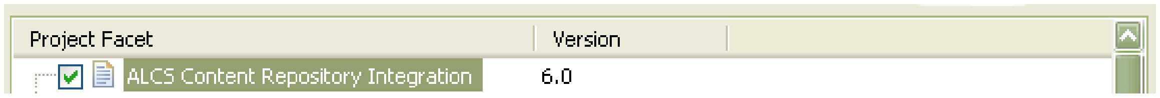
2. Click on “Project Facets”



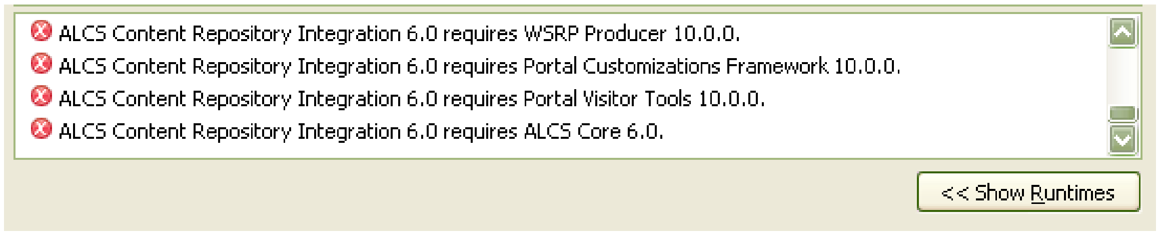
3. Click on “Add/Remove Project Facets…”



4. Select the “ALCS Content Repository Integration” facet item



5. If you have required facets to add along with “ALCS Content Repository Integration”, you will see a warning message at the bottom of the screen:


ALCS Content Repository Integration 6.0 requires WSRP Producer 10.0.0.
ALCS Content Repository Integration 6.0 requires Portal Customizations Framework 10.0.0.
ALCS Content Repository Integration 6.0 requires Portal Visitor Tools 10.0.0.
ALCS Content Repository Integration 6.0 requires ALCS Core 6.0.

Make sure to select all Facets listed on the Warning box.

6. After selecting the “ALCS Content Repository Integration” facet and its dependency facets, click Next
7. On the next screen, select your preferred WebLogic Share J2EE Library deployment.

   Note: AquaLogic Commerce Services libraries are copied to either your Enterprise Application or your Web Application, independently of the option you select on this screen.
8. Click Finish and wait until all libraries are copied to your Enterprise Application.

Installing Manually

Manual deployment is also an option to perform the deployment of the Content Management integration.

In order to deploy manually the Content Management integration, copy the following files to the specified location on your WebLogic Portal Enterprise Application.

- ENTERPRISE_APPLICATION/APP-INF/lib
- The files from WEBLOGIC_HOME/commerce/samples/libs/repo and WEBLOGIC_HOME/commerce/samples/libs/repo/dependencies
- ENTERPRISE_APPLICATION/META-INF
- The files **content-config.xml** and **p13n-cache-config.xml** from WEBLOGIC_HOME/commerce/samples/libs/repo/xml

How to Uninstall from Your Project

Removing the “ALCS Content Repository Integration” facet from your application, will not remove the previously copied files to your WebLogic Portal Enterprise Application.

In order to completely remove the Content Management integration component, you will need to delete manually:

- The files within ENTERPRISE_APPLICATION/APP-INF/lib that are not required by your application;
- Remove the repository configuration from ENTERPRISE_APPLICATION/META-INF/content-config.xml
- Remove the cache configuration from ENTERPRISE_APPLICATION/META-INF/p13n-cache-config.xml

Configuring Repository Connection Information

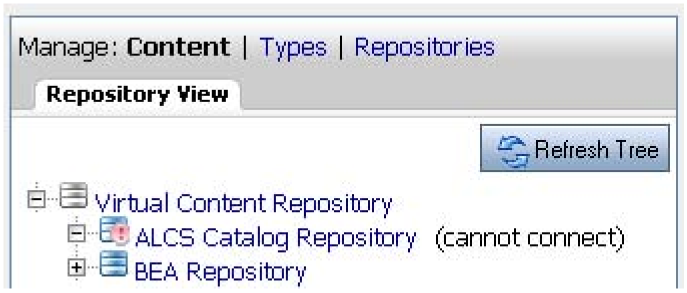
In order to connect to the Commerce Manager server and retrieve correct information, you have several repository attributes that you should configure prior to usage. These attributes are detailed on the next sections. However, if you have added the Content Management integration through the WorkSpace Studio Facet, all the attributes have already been added to your content-config.xml descriptor file, and you should now only configure the values for them through the Portal Administration Console.

Configuring through the Portal Administration Console

The new Content Repository configuration will be made available right after the deployment of your application.

If you will add the Content Management integration manually and have not yet added the attributes, you should add them using the Portal Administration Console. If you need more information on how to configure Repository attribute, please check BEA eDocs website.

Until you have properly configured the Repository attributes, you will see your Repository information as below:


Manage: Content | Types | Repositories
Repository View
Refresh Tree
Virtual Content Repository
ALCS Catalog Repository (cannot connect)
BEA Repository

Available Configuration Attributes

The following attributes are available and require configuration:

Attribute Name	Description
name	The Repository name you want to use. This will be used during retrieval of products and/or categories by path.

	For example, if you configure ALCS MyStore, your paths will start with: /ALCS MyStore/
description	The description of your Repository.
class-name	The SPI class that implements the Repository interface as specified by WebLogic Portal Framework. To connect to an AquaLogic Commerce Services catalog, the class that should be used is: com.bea.alcs.spi.RepositoryImpl
username	The username used to connect to your Commerce Manager Server. This value should not be encrypted.
Store Code	The Store Code you want this Repository to retrieve products and categories from. If you have more than one store configure on your AquaLogic Commerce Services installation and want to access by your WebLogic Portal application, you should create another repository on either content-config.xml or through Portal Administration Console, and change the Store Code to the new store you want to use. A single Repository configuration CANNOT and SHOULD NOT retrieve products from more than one store.
FrontEnd Address	The full HTTP or HTTPS to your Commerce Manager Server. If connection is not available, your Content Repository will not be available to retrieve products and categories from. Make sure your Commerce Manager Server is accessible by all your WebLogic Portal servers using the Content Management services. For more information on Commerce Manager Server, please check BEA AquaLogic Commerce Services documentation at BEA eDocs website.
password-encrypted	The password used to connect to the Commerce Manager Server. If the username and password are both entered through the portal administrator, the password will be encrypted inside this file. If the password is to be entered manually, the password must be encrypted to this domain's key, using the weblogic.security.Encrypt, as described in: http://e-docs.bea.com/wls/docs100/admin_ref/utils.html#wp1209592

Configuring Cache Information

The Content Management integration leverages all WebLogic Portal framework caching functionality, giving you the same configuration options and facilities as the Out-of-the-box Repository services.

Any of the caches defined by the application can be managed just like standard weblogic caches, as described in: http://e-docs.bea.com/wlp/docs100/perftune/4PortalApplication.html#wp1072892

The file **p13-cache-config.xml** is responsible for the default values the cache will have upon application deployment. These caches have a big impact on the performance and freshness of the data made available through SPI.

The following cache types are available:

repo.type.<repository name>	Cache of the object model that defines the different metadata for each product type.
repo.product.<repository name>	Cache of the product nodes containing all products attributes, including relationships with other products.
repo.category.<repository name>	Cache of the category nodes containing categories names and paths. Attributes are not cached.

<repository name> should be replaced by the name of the repository you are configuring to connect to the AquaLogic Commercer Services Manager Server. This is to avoid the situation whereas you configure several repositories to connect to different configured stores on your ALCS installation.

For the example, the three cache names above would be named, in case the repository name is "ALCS Catalog Repository":

repo.type.ALCS Catalog Repository

repo.product.ALCS Catalog Repository

repo.category.ALCS Catalog Repository

The cache can be configured in two ways:

- Descriptor file: configure your cache information using the p13-cache-config.xml descriptor file, located on the META-INF directory of your WebLogic Portal Enterprise Application;
- Portal Administration Console: it is also possible to configure on-the-fly your cache by using the Portal Administration Console. Configuring through the Portal Administration Console also gives you the possibility of flushing the entire cache or by a specific key. For example, you would want to remove outdated product information from the cache right after you have updated its information through the Commerce Manager application.

Changes needed for new products types or categories

There is no need to alter the SPI in any way when adding product type or adding new categories. When these changes are done in the CM Client tools, they will be automatically captured by the SPI implementation and will be made available.

WebLogic Portal Sample Portlets

Out of the box in AquaLogic Commerce Services, BEA provides you three sample portlets to demonstrate the many ways you can leverage the information from ALCS within your WebLogic Portal. These portlets are current shopping cart, order history and cross-sell/up-sell.

- Shopping Cart – Lets users view their current shopping cart and provides a link to let the user return to the full storefront shopping cart.
- Cross-Sell/Up-Sell – Shows the user products he might also be interested in purchasing. The “you may also like” products are populated based on a) shopping cart contents, b) order history, or c) both shopping cart and order history. Clicking a product name returns the user to the store.
- Order History – Lets the user view their order/purchase history. Users can click a link to go to the self-service area of the storefront.

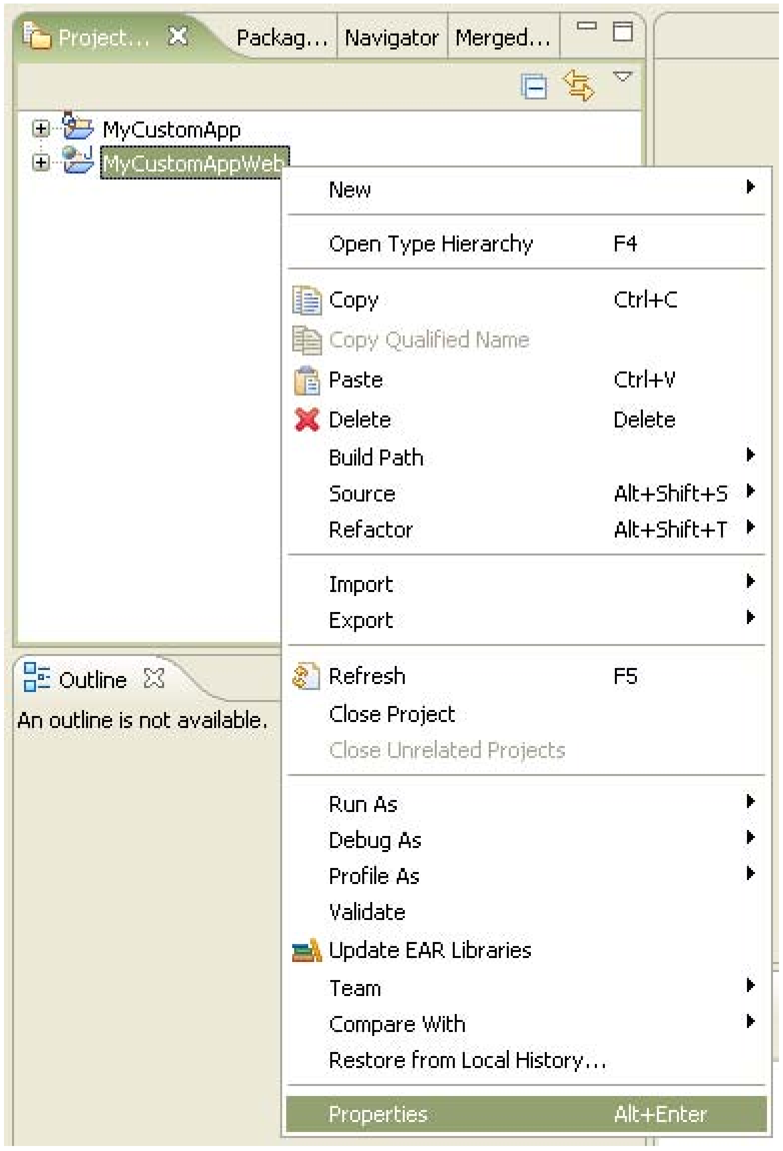
Adding the Sample Portlets to your Web Application

Adding by using WorkSpace Studio Facet

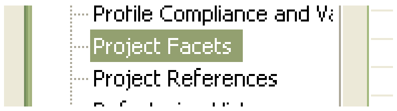
This is the recommended way to add Content Management integration to your application as it will add for you automatically all required libraries and make the proper configurations on your WebLogic Portal Enterprise Application.

To add the Content Management integration Facet to your WebLogic Portal Enterprise Application, execute the following procedures:

1. Right-Click on your WebLogic Portal Web Application and select **Properties**

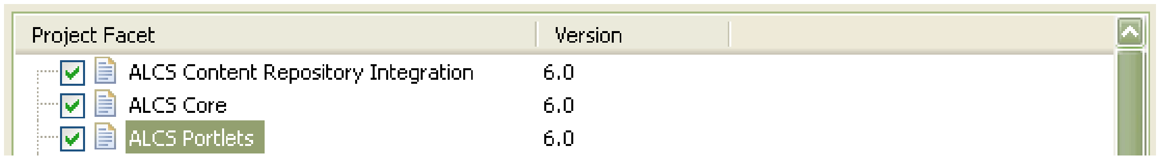


2. Click on “Project Facets”



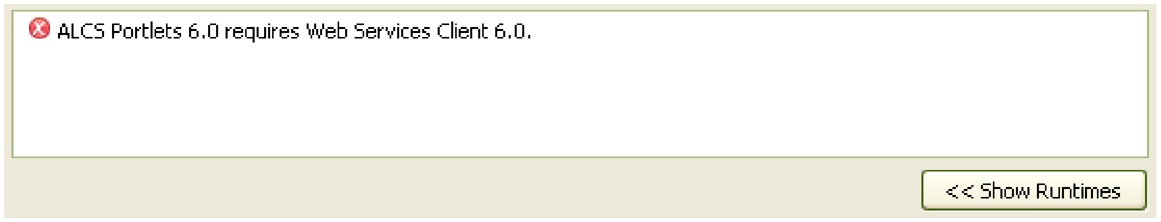
3. Click on “Add/Remove Project Facets…”

4. Select the “ALCS Portlets” facet item



Project Facet	Version
✓ ALCS Content Repository Integration	6.0
✓ ALCS Core	6.0
✓ ALCS Portlets	6.0

5. If you have required facets to add along with "ALCS Portlets", you will see a warning message at the bottom of the screen:


ALCS Portlets 6.0 requires Web Services Client 6.0.

   Make sure to select all Facets listed on the Warning box.
6. After selecting the "ALCS Portlets" facet and its dependency facets, click Next
7. On the next screen, select your preferred WebLogic Share J2EE Library deployment.

   Note: AquaLogic Commerce Services libraries are copied to either your Enterprise Application or your Web Application, independently of the option you select on this screen.
8. Click Finish and wait until all java source, pageflows, JSPs and required libraries are copied to your both Enterprise and Web Application.

Configuring Connection Information

After successfully adding the sample portlets to your web application, you need to configure the connection information used by the WebServices Client to connect to the WebServices Server, in order to retrieve product, shopping cart, customer and order information.

To configure you connection information, search for the file named "CommerceWSServer.properties" inside your source folder. There are three properties that you will need to configure:

WSServerUrl	The full URL to your AquaLogic Commerce Services WebServices Server.
WSUsername	The username configured on AquaLogic Commerce Services with permission configured to use WebServices.
WSPassword	The password of the above mentioned user.

Tip: Connection is managed through the class com.bea.alcs.webservice.WebServiceBase

Tip: WebServices usage are wrapped on the classes inside the package com.bea.alcs.webservice

Shopping Cart Portlet

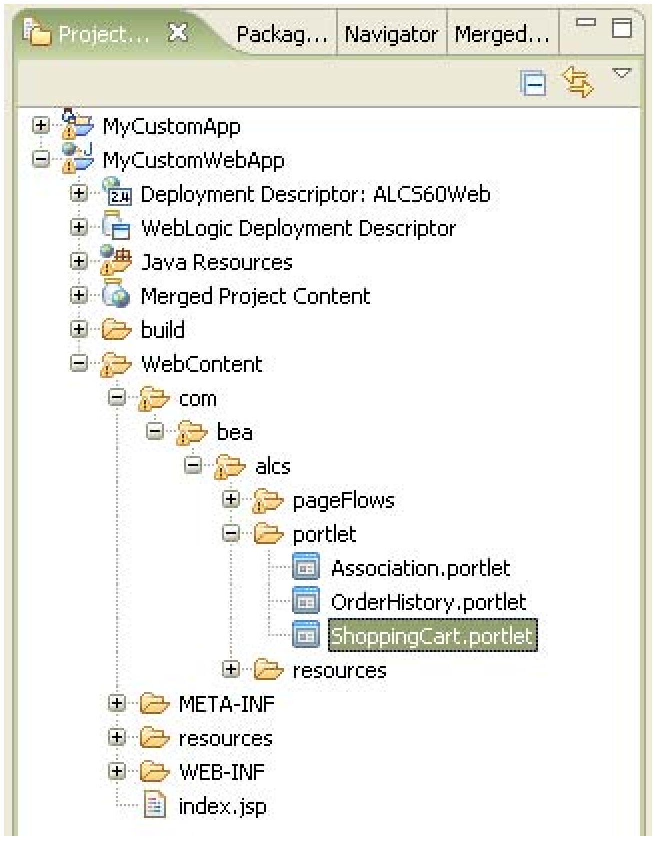
The Shopping Cart Portlet provides you all source code example on using two WebServices and the Content Management JSP Tags.

First, it retrieves the customer information through the CustomerWebService using the logged on user. Later it retrieves the product SKUs currently on the user's Shopping Cart for currently configured Store Code on the portlet's preference CurrentStoreCode. The Shopping Cart information is retrieved using the ShoppingCartWebService.

When Product SKUs are retrieved from the WebService, the product information (including its name), sale price, product image, and some other information is retrieved using the Content Management JSP Tags.

You can find the Shopping Cart portlet at:

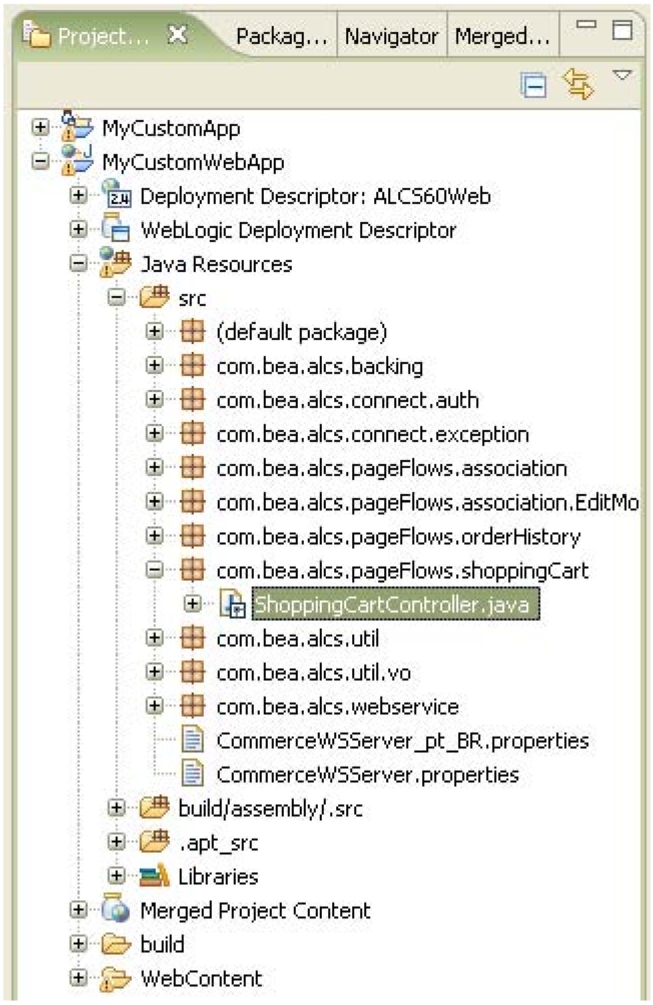
WEB_APPLICATION/WebContent/com/bea/alcs/portlet/ShoppingCart.portlet



```
MyCustomApp
MyCustomWebApp
  Deployment Descriptor: ALCS60Web
  WebLogic Deployment Descriptor
  Java Resources
  Merged Project Content
  build
  WebContent
    com
      bea
        alcs
          pageFlows
          portlet
            Association.portlet
            OrderHistory.portlet
            ShoppingCart.portlet
          resources
    META-INF
    resources
    WEB-INF
    index.jsp
```

You can find the Shopping Cart Pageflow at:

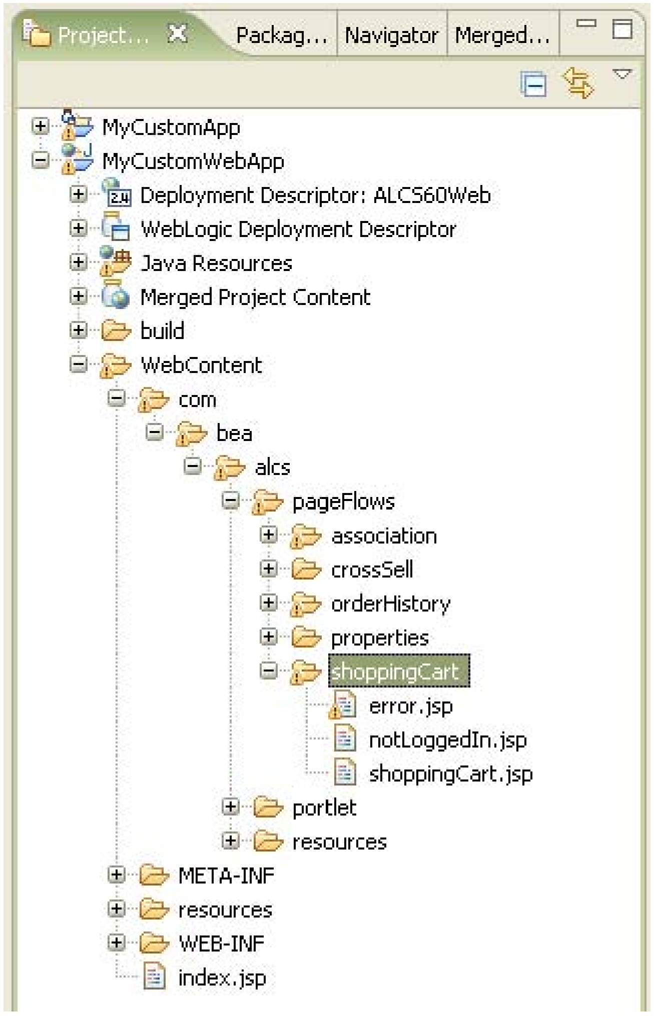
WEB_APPLICATION/Java Resources/src/com.bea.alcs.pageFlows.shoppingCart



```
MyCustomApp
MyCustomWebApp
  Deployment Descriptor: ALCS60Web
  WebLogic Deployment Descriptor
  Java Resources
    src
      (default package)
      com.bea.alcs.backing
      com.bea.alcs.connect.auth
      com.bea.alcs.connect.exception
      com.bea.alcs.pageFlows.association
      com.bea.alcs.pageFlows.association.EditMo
      com.bea.alcs.pageFlows.orderHistory
      com.bea.alcs.pageFlows.shoppingCart
        ShoppingCartController.java
      com.bea.alcs.util
      com.bea.alcs.util.vo
      com.bea.alcs.webservice
      CommerceWSServer_pt_BR.properties
      CommerceWSServer.properties
    build/assembly/.src
    .apt_src
    Libraries
  Merged Project Content
  build
  WebContent
```

You can find the Shopping Cart JSPs at:

WEB_APPLICATION/WebContent/com/bea/alcs/pageFlows/shoppingCart

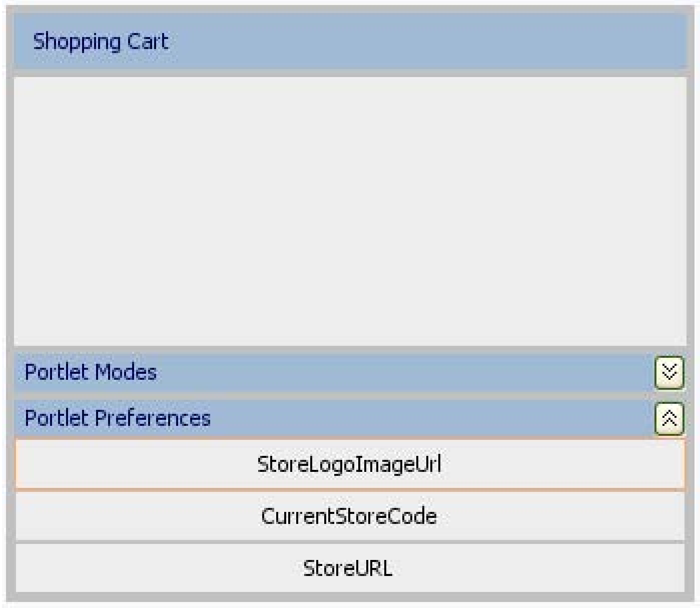


```
MyCustomApp
MyCustomWebApp
    Deployment Descriptor: ALCS60Web
    WebLogic Deployment Descriptor
    Java Resources
    Merged Project Content
    build
    WebContent
        com
            bea
                alcs
                    pageFlows
                        association
                        crossSell
                        orderHistory
                        properties
                        shoppingCart
                            error.jsp
                            notLoggedIn.jsp
                            shoppingCart.jsp
                    portlet
                    resources
        META-INF
        resources
        WEB-INF
        index.jsp
```

Portlet Preferences

The Shopping Cart Portlet has three preferences that you need to configure before using it on your WebLogic Portal Web Application.

For more information on configuring portlet preferences within WorkSpace Studio and Portal Administration Console, please refer to BEA eDocs website.



StoreLogoImageUrl	The full URL where your store logo image is located. It should be on a server accessible by the users you expect to use this portlet.
CurrentStoreCode	The Store Code you want to limit the products from. Only products from the selected Store will be visible, even if the user has products on more than one Shopping Cart on different stores.
StoreURL	The URL to the main web page of your store, or the URL you want your user to go when clicking on the Store Logo image, and the “Go To Store!” links.

Order History Portlet

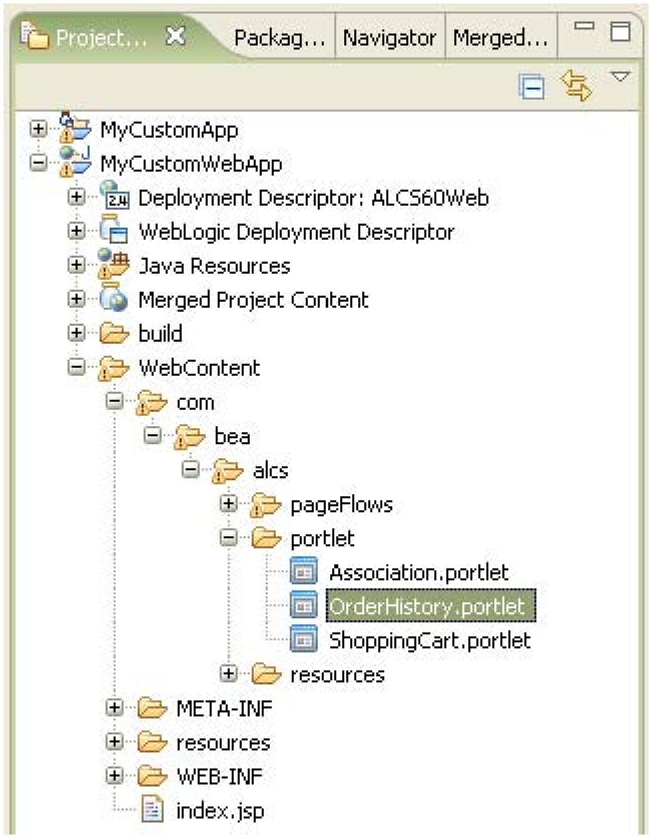
The Order History Portlet provides you all source code example on using two WebServices and the Content Management JSP Tags.

First it retrieves the customer information through the CustomerWebService using the logged on user. Later it retrieves all order history, along with its product SKUs for the currently configured Store Code on the portlet’s preference CurrentStoreCode. The Order History information is retrieved using the OrderWebService.

When order history is retrieved from the WebService, the product name is retrieved using the Content Management JSP Tags.

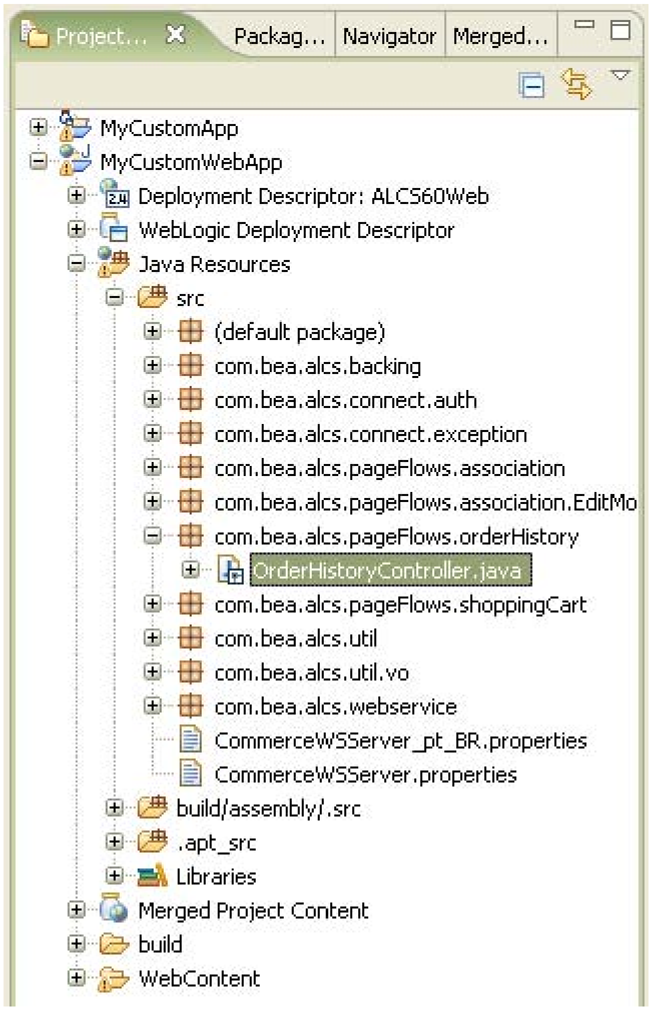
You can find the Order History portlet at:

WEB_APPLICATION/WebContent/com/bea/alcs/portlet/OrderHistory.portlet



You can find the Order History Pageflow at:

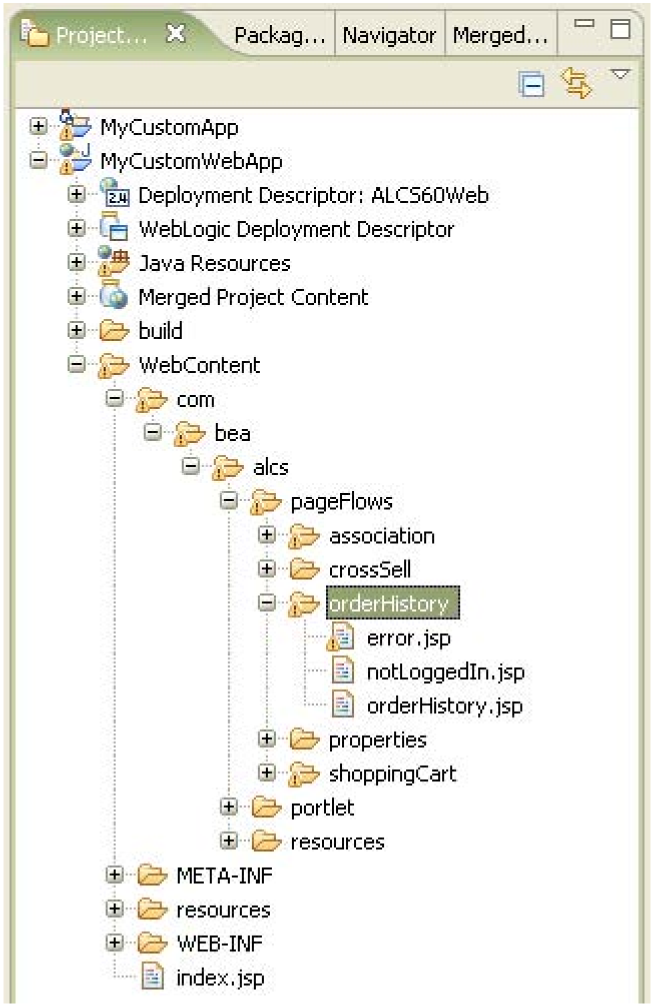
WEB_APPLICATION/Java Resources/src/com.bea.alcs.pageFlows.orderHistory



```
MyCustomApp
MyCustomWebApp
    Deployment Descriptor: ALCS60Web
    WebLogic Deployment Descriptor
    Java Resources
        src
            (default package)
            com.bea.alcs.backing
            com.bea.alcs.connect.auth
            com.bea.alcs.connect.exception
            com.bea.alcs.pageFlows.association
            com.bea.alcs.pageFlows.association.EditMo
            com.bea.alcs.pageFlows.orderHistory
                OrderHistoryController.java
            com.bea.alcs.pageFlows.shoppingCart
            com.bea.alcs.util
            com.bea.alcs.util.vo
            com.bea.alcs.webservice
            CommerceWSServer_pt_BR.properties
            CommerceWSServer.properties
        build/assembly/.src
        .apt_src
        Libraries
    Merged Project Content
    build
    WebContent
```

You can find the Order History JSPs at:

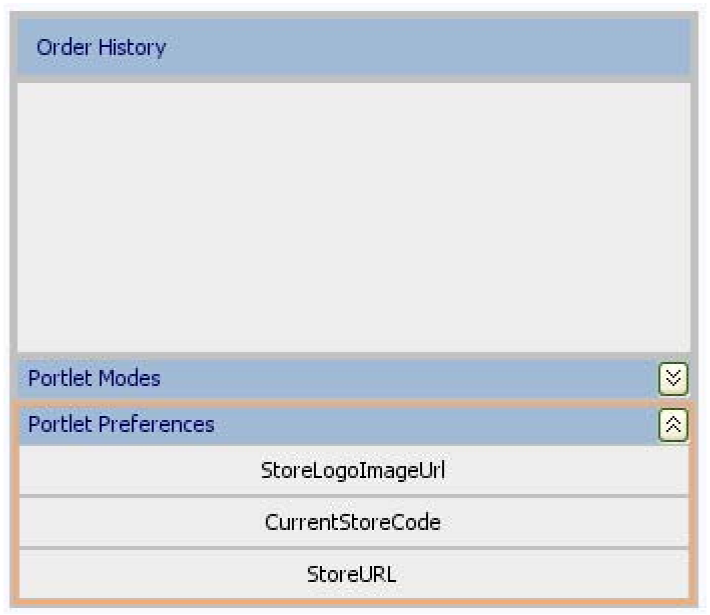
WEB_APPLICATION/WebContent/com/bea/alcs/pageFlows/orderHistory



Portlet Preferences

The Order History Portlet has three preferences that you need to configure before using it on your WebLogic Portal Web Application.

For more information on configuring portlet preferences within WorkSpace Studio and Portal Administration Console, please refer to the BEA eDocs website.



StoreLogImageUrl	The full URL where your store logo image is located. It should be on a server accessible by the users you expect to use this portlet.
CurrentStoreCode	The Store Code you want to limit the products from. Only products from the selected Store will be visible, even if the user has order history at other stores.
StoreURL	The URL to the main web page of your store, or the URL you want your user to go when clicking on the Store Logo image, and the "Go To Store!" links.

Portlet preference URL's

When inputing URL's for portlet preferences in testing and production, make sure that they match the URL used when logging into the storefront.

i.e. If you login on the Storefront using http://localhost:7011/, you cannot try to use the portlets using http://somedomain:7011/, since the JSESSIONID cookie is registered to domain 'localhost'. When you access as http://somedomain:7011/, WLS will look for the cookies registered for domain 'somedomain'. This is a security measure applied by web browsers.

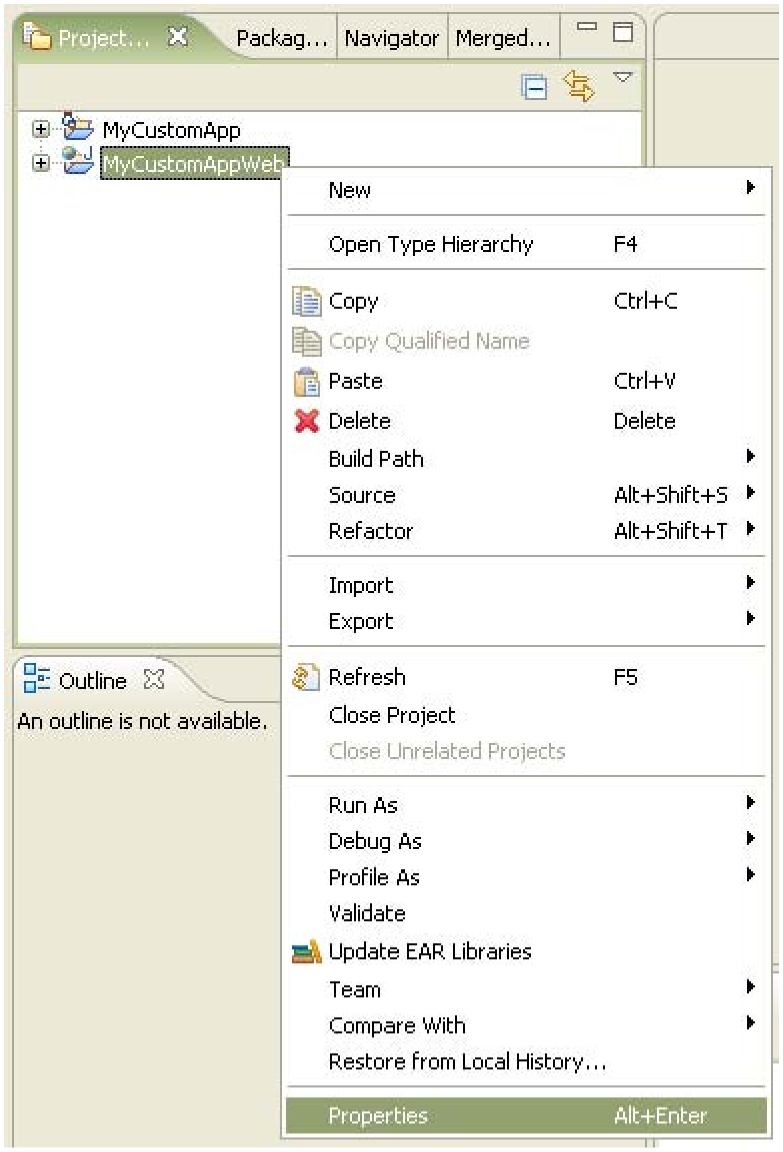
Adding AquaLogic Commerce Services Core Components

Adding by WorkSpace Studio Facet

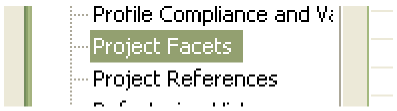
This is the recommended way to add Core components to your application, as it will add for you automatically all required libraries and its dependencies to either your WebLogic Portal Enterprise application or your WebLogic Portal Web application.

To add the ALCS Core facet to your WebLogic Portal Enterprise or Web application, follow these steps:

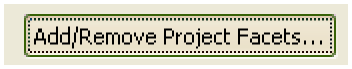
1. Right-click on your WebLogic Portal Enterprise or Web application and select Properties



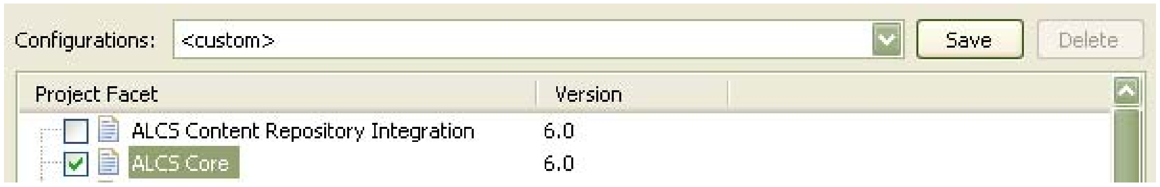
2. Click on “Project Facets”



3. Click on "Add/Remove Project Facets…"



4. Select the "ALCS Core" facet item



Configurations: <custom>

Project Facet	Version
☐ ALCS Content Repository Integration	6.0
✓ ALCS Core	6.0

5. After selecting the "ALCS Core" facet, click Next
6. On the next screen, select your preferred WebLogic Share J2EE Library deployment if required.

   Note: AquaLogic Commerce Services libraries are copied to either your Enterprise Application or your Web Application, independently of the option you select on this screen.
7. Click Finish and wait until AquaLogic Commerce Services Core components and its dependencies libraries are copied to your WebLogic Portal Enterprise or Web Application.

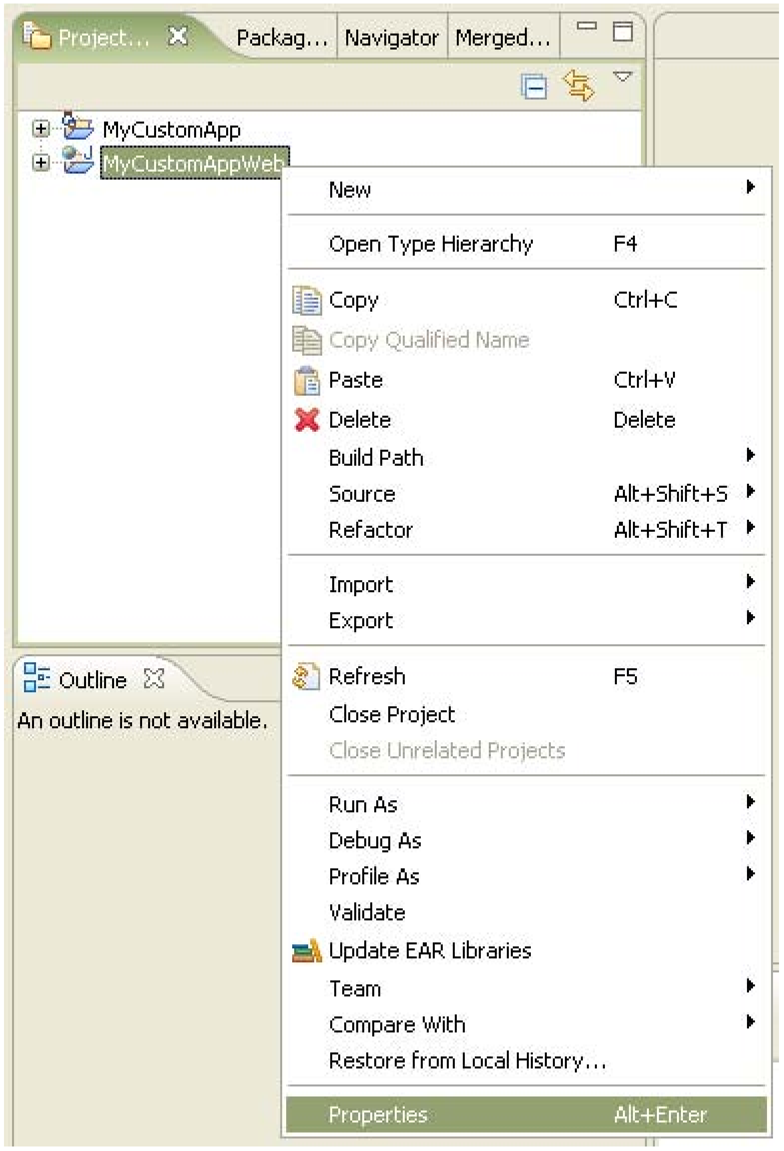
Adding AquaLogic Commerce Services Web Services

Adding by WorkSpace Studio Facet

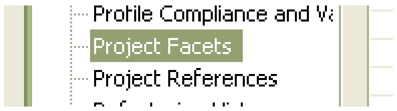
This is the recommended way to add the WebServices Client to your application, as it will add for you automatically all required libraries and its dependencies to either your WebLogic Portal Web application.

To add the ALCS WebServices Client facet to your WebLogic Portal Web application, fo9llow these steps:

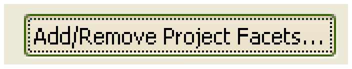
1. Right-click on your WebLogic Portal Web application and select Properties



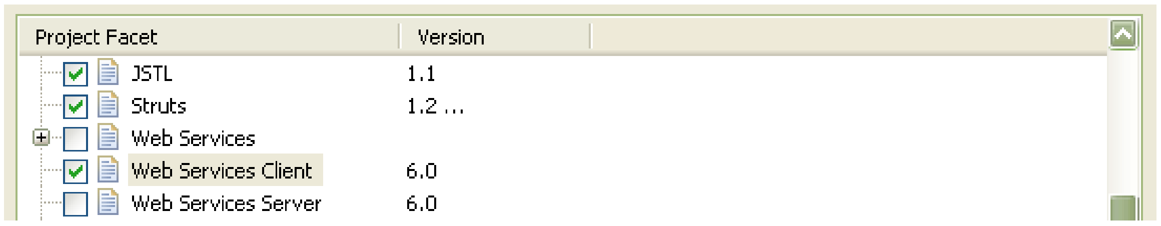
2. Click on “Project Facets”



3. Click on “Add/Remove Project Facets…”



4. Select the “ALCS WebServices Client” facet item



Project Facet	Version
✓ JSTL	1.1
✓ Struts	1.2 ...
Web Services	
✓ Web Services Client	6.0
Web Services Server	6.0

5. After selecting the “ALCS WebServices Client” facet, click Next
6. On the next screen, select your preferred WebLogic Share J2EE Library deployment if required.

   Note: AquaLogic Commerce Services libraries are copied to your Web Application, independently of the option you select on this screen.
7. Click Finish and wait until AquaLogic Commerce Services WebServices Client components and its dependencies libraries are copied to your WebLogic Portal Web Application.

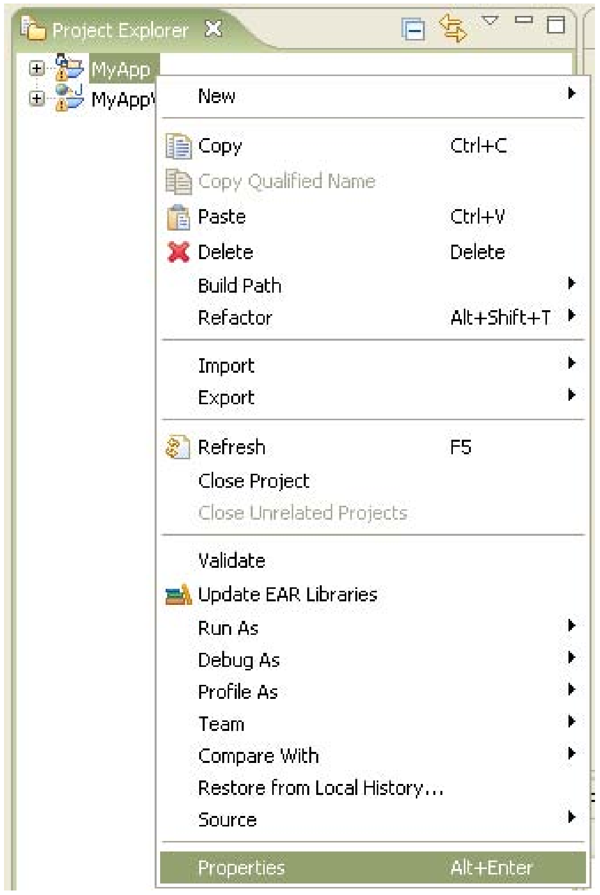
Adding by WorkSpace Studio Facet

Note: The description below asks to add the facet to the EAR Project. The current version only shows the WebServices Server on the Web Application. This will be changed on the next Facets release. Also the screenshots will be changed.

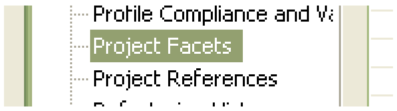
This is the recommended way to add the WebServices Server to your application, as it will add for you automatically the application WAR file to your Enterprise application.

To add the ALCS WebServices Server facet to your WebLogic Portal Enterprise application, execute the following procedures:

1. Right-click on your WebLogic Portal Enterprise application and select Properties

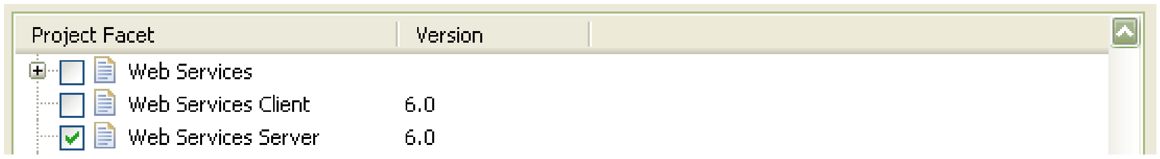


2. Click on “Project Facets”



3. Click on “Add/Remove Project Facets…”

4. Select the “ALCS WebServices Server” facet item



Project Facet	Version
☐ Web Services	
☐ Web Services Client	6.0
✓ Web Services Server	6.0

5. After selecting the “ALCS WebServices Server” facet, click Next
6. Click Finish and wait until AquaLogic Commerce Services WebServices Server WAR application is copied to your WebLogic Portal Enterprise Application.

Search Capabilities

You can leverage all the search capabilities available on BEA WebLogic Portal by using the Content Management APIs and JSP tags, building your own search expressions, and searching for products and categories.

The returned Node objects will carry all attributes, except the ones returned by getProperties() method. The getProperties() call is lazy-loaded, going back to your Commerce Manager Server to retrieve your customized and localizable attributes. For more information on managing customized attributes and localization, please check BEA AquaLogic Commerce Services documentation at BEA eDocs website.

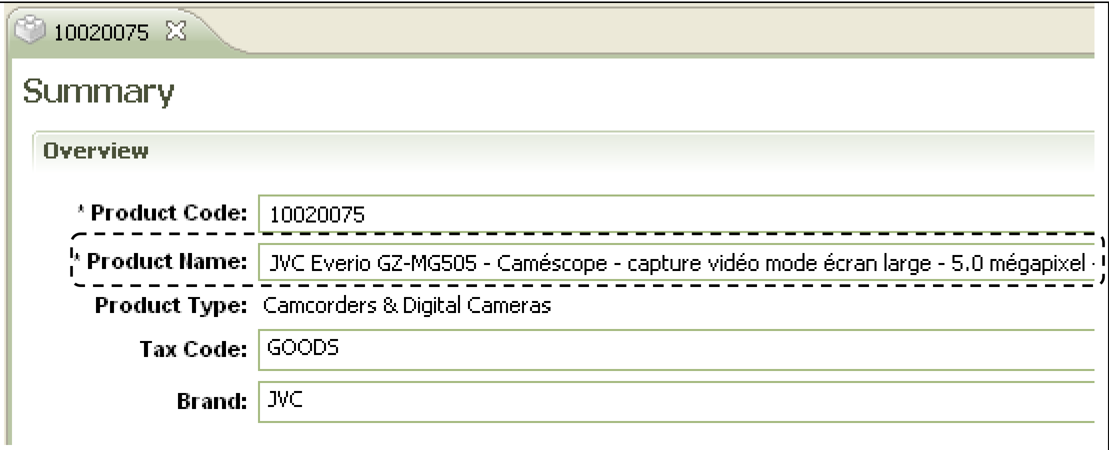
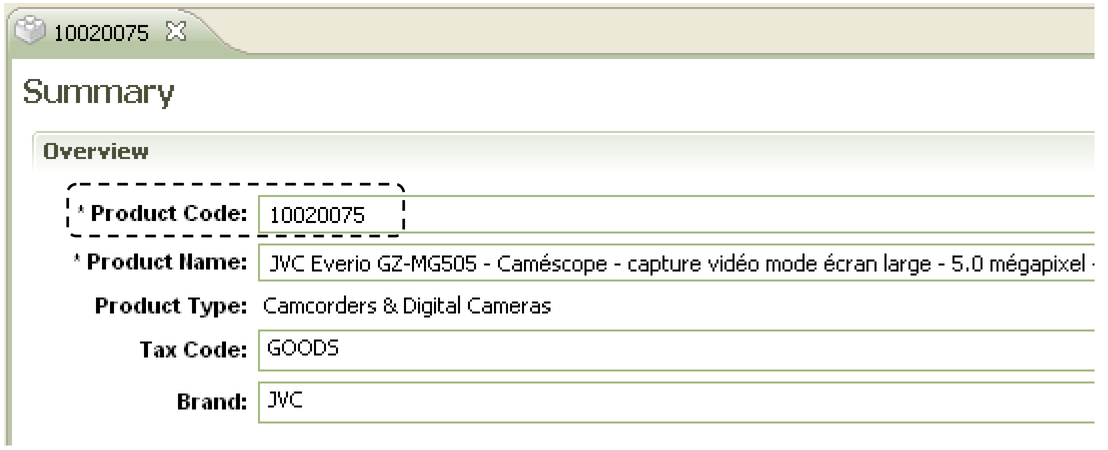
For more information on using the Content Management APIs, building Expressions and many other options, please refer to the BEA WebLogic Portal documentation at BEA eDocs website.

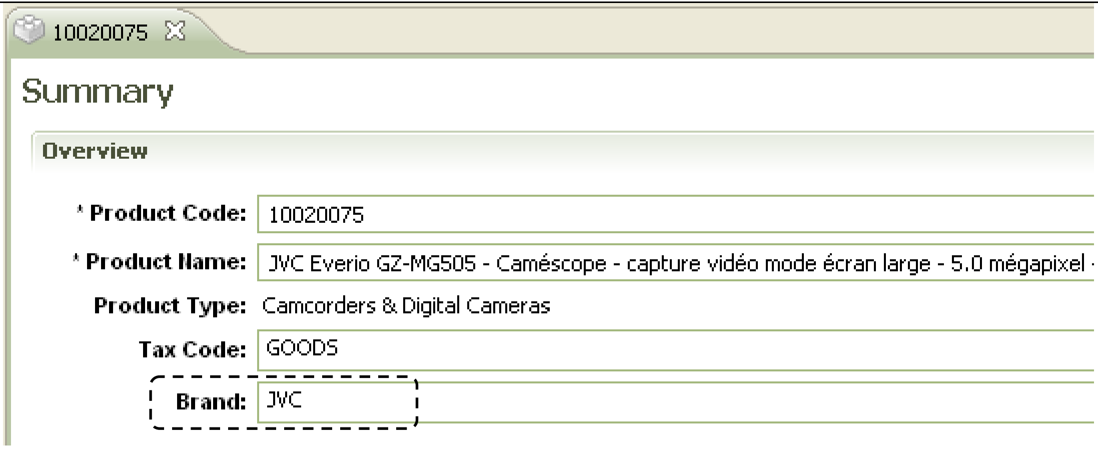
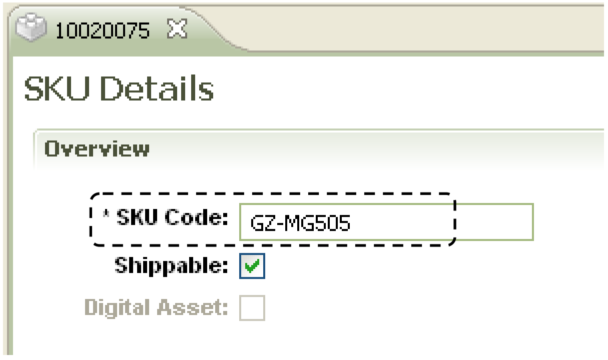
Available search attributes and conditions

A set of attributes is available for usage on your expressions and only EQUAL and STRINGLIKE comparative operators are available. If other attributes or comparative operators are used, a RepositoryException is thrown indicating that you have tried to use an invalid attribute or operator.

The search attributes available are:

cm_nodeName	The name of the product or the category you want to search for.

	 10020075 Summary Overview * Product Code: 10020075 * Product Name: JVC Everio GZ-MG505 - Caméscope - capture vidéo mode écran large - 5.0 mégapixel · Product Type: Camcorders & Digital Cameras Tax Code: GOODS Brand: JVC
cm_uid	The Product Code or the Category Code you want to search for.  10020075 Summary Overview * Product Code: 10020075 * Product Name: JVC Everio GZ-MG505 - Caméscope - capture vidéo mode écran large - 5.0 mégapixel · Product Type: Camcorders & Digital Cameras Tax Code: GOODS Brand: JVC
cm_category	You can restrict your searches for product based on specific categories they are associated with. This search attribute also restricts your searches for categories that are child of the specified category. The value used here should be the Category Code.
cm_locale	You can restrict your product or category searches based on the specified locale. The values are based on a two-digit character. For example, for English US: us
cm_currency	The currency you want to restrict you product search with. Only products that have the specified currency configured will be returned.
cm_brand	The brand(s) you want to restrict your product search with. Only products that have the specified brand associated with will be returned.

	 10020075 Summary Overview * Product Code: 10020075 * Product Name: JVC Everio GZ-MG505 - Caméscope - capture vidéo mode écran large - 5.0 mégapixel Product Type: Camcorders & Digital Cameras Tax Code: GOODS Brand: JVC
cm_sku	Search for a product with the specified SKU.  10020075 SKU Details Overview * SKU Code: GZ-MG505 Shippable: ✓ Digital Asset:

Example Search Expressions

Performing search operations on an AquaLogic Commerce Services catalog is as seamless as searching the out-of-the-box WebLogic Portal Content Repository.

Search Expressions on the Content Management are SQL-alike and can have any number of combinations to retrieve the exact products and/or categories you want to use on your WebLogic Portal application.

Some example expressions for searching categories and products are provided below. For more information on search expressions and other WebLogic Portal Content Management capabilities, please refer to the BEA eDocs website.

Searching products and categories by name

cm_nodeName == ‘Canon Rebel D100 / Memmory Card SD’

cm_nodeName likeignorecase ‘Canon*’

cm_nodeName likeignorecase ‘Canon*’ || cm_nodeName likeignorecase ‘Memory*’

Searching products by Brand

cm_brand == ‘Sony’

cm_brand == ‘Sony’ || cm_brand == ‘JVC’

Searching products by SKU

cm_sku == ‘CAN1309TG’

Searching by name and product within a specified category

cm_nodeName likeignorecase ‘Memory*’ && cm_category == ‘MEM309LK’

The Forward Slash (/) within Category and Product Names

Since the forward slash symbol (/) is used as the separator on a Content Node path, when products or categories names are retrieved, the forward-slash symbols, if found, are replaced by a hyphen symbol (“-“).

For example, if the product name ingested on AquaLogic Commerce Services is:

“Canon Rebel D100 / Memory Card SD”

When the Node is returned, the name attribute will be:

“Canon Rebel D100 - Memory Card SD”

This replacement is required to avoid impacts on Content Node path that is composed by the Repository Name, the root and its child categories, and the product name.

For search expressions, you can use the forward slash as usual. There is no need to replace with the hyphen symbol.

⚠ **Placeholders and Content Selectors**

This functionality is available, but not available for preview when working in Workshop.

⚠ **Full text search**

This functionality is not available in this release of AquaLogic Commerce Services.

Integrating with Back-End Systems

AquaLogic Commerce Services includes support for integrating your AquaLogic Commerce Services store with external systems.

AquaLogic Commerce Services Web Services

AquaLogic Commerce Services web services layer provides external systems programmatic access to the application using a simple, powerful, and secure application programming interface. AquaLogic Commerce Services web services exposed business functionality for customer service, order management, and other other commerce functionality.

For more information please refer to the AquaLogic Commerce Services Web Services documentation in the *AquaLogic Commerce Services Developer Guide.*

Integrating Business Process Management (BPM) Functionality

AquaLogic Commerce Services can be used with AquaLogic Business Process Management (BPM). Using the AquaLogic BPM Suite you can easily register and consume Web services, such as the Web services provided by the AquaLogic Commerce Services Web Services module.

You can also integrate with AquaLogic Service Bus by configuring and registering AquaLogic Commerce Services Web Services with the service bus, configuring its related proxy services, and exposing them as Web services. AquaLogic Service Bus can help configuration and deployment, and it simplifies management of shared services across a SOA environment.

For more information see the AquaLogic BPM documentation at http://edocs.bea.com/albsi/docs57/index.html the AquaLogic Service Bus documentation at http://edocs.bea.com/alsb/docs26/index.html and the AquaLogic Commerce Services Web Services documentation.

Setting up an FTP Server

This section covers setting up an FTP server on the AquaLogic Commerce Server, to handle files for the Commerce Manager clients. Files transferred with the FTP server include product images and import files.

Installing an FTP Server

If you do not already have an FTP server installed on the server upon which the Commerce Server is installed, you must install one. There are literally dozens of FTP server software packages available; however, if you are using Linux, you may want to consider ProFTPd. If your server is running Windows, popular FTP programs include CrossFTP and FileZilla.

Configuring the FTP server

1. Create a new FTP user for the Commerce Server clients to use (e.g. commerce)
2. The FTP user's Home Directory should ideally be the assets folder that has been configured in the Commerce Server's commerce-config.xml file, in the property "catalog.asset.path". If it is a different folder, the "asset.vfs.rootpath" should be set to the absolute path of the asset folder. In this case you should ensure your ftp server is configured to make this path accessible to your ftp user.
3. Verify that the FTP user has read/write permissions on the assets folder and its subfolders.

Configuring Commerce Server

The Commerce Server's commerce-config.xml file is downloaded to the Commerce Manager client (and kept in memory only) when the user logs in. Relevant properties in the file that must be modified to fit your installation are in the <asset> element:

```
<property name="catalog.asset.path" value="/opt/commerce/assets"/>
<property name="image.asset.subfolder" value="images"/>
<property name="file.asset.subfolder" value="files"/>
<property name="import.asset.subfolder" value="import"/>
<property name="digitalgoods.asset.subfolder" value="digitalassets"/>
<!-- Used by the Commerce Manager Clients -->
<property name="asset.vfs.protocol" value="ftp"/>
<property name="asset.vfs.host" value="my.commerce.server"/>
<property name="asset.vfs.port" value="21"/>
<property name="asset.vfs.username" value="commerce"/>
<property name="asset.vfs.password" value="commerce"/>
<property name="asset.vfs.rootpath" value=""/>
<property name="asset.http.host" value="http://my.commerce.server/cmserver"/>
```

While ftp is the officially supported protocol for the Commerce Manager clients to use, we are using Apache Commons VFS, which allows other protocols. You can set the "asset.vfs.protocol" to alternatives such as "sftp" for SFTP.

Developer (Single box) Installations

If you are installing the Commerce Server on the same machine upon which you are running the Commerce Manager Client, and intend to run any import tasks or upload any images that use the FTP capabilities, you must be careful not to upload files from one directory into the same directory. Developers may wish to set this to "file", and change the rootpath to point to the destination top-level assets directory on their local drive. This would avoid requiring that an FTP server be set up in development environments.

Troubleshooting

This section outlines some of the common problems that our customers have run into while attempting to deploy AquaLogic Commerce Services.

Errors executing database queries

Symptom:

The applications start up, but the Commerce Manager presents a notice screen stating "Request could not be fulfilled" or the Storefront displays error "Please contact the store administrator".

Solution:

Different application servers sometimes require different JNDI resource reference strings. Ensure that you are using the correct build for the application server that you are using because the scripts that create the build will ensure that the string is formatted correctly for that application server.

Symptom (Specific to MS SQL Server 2005):
Receive the following error in the logs:
com.microsoft.sqlserver.jdbc.SQLServerException: sp_cursoropen/sp_cursorprepare: The statement parameter can only be a batch or a stored procedure with a single select, without FOR BROWSE, COMPUTE BY, or variable assignments.

Solution:
When using MS SQL Server 2005, be sure to download the sqljdbc 1.1 version rather than the sqljdbc 1.0 version of the JDBC driver. For more information, see **JNDI and JDBC Configuration** section above.

The Storefront is running, but no products are showing up under the categories

Symptoms:
In the storefront, the index page loads and the default category tabs show, but there are no products in the categories. When you click on one of the products or categories on the index page, you see one of the following error messages:

- "The request cannot be fulfilled successfully. Please contact the store administrator."
- "The catalog view request is invalid."
- "Sorry, this product is no longer available in the store."

Solutions:

- Running the supplied SQL scripts alone does not populate the demo store with products. To populate the store with our demo SnapItUp products, run the import jobs using the Import tool in the Commerce Manager. Step-by-step instructions can be found here in the **Running SnapItUp Import Jobs (optional)** section above.

- It is possible that the products have been populated but there is no inventory for the products. In that case, you can log in to the Commerce Manager Client and search for the products there to determine whether they have any inventory.
- If this is a new store, you will need to rebuild your indices before the products will appear:
    - Go to the …**\webapps\searchserver\WEB-INF\solrHome** folder and delete the **data** folder from that directory.
    - Go to the **...\webapps\searchserver\WEB-INF\conf\resources** folder. Open each properties file for each index type and delete the timestamp next to LastBuildDate=. This will cause the index to rebuild.

Portlets do not show my products

Symptom:
User is registered in the storefront, but nothing shows up in the ShoppingCart and OrderHistory portlets for the user.

Solution:
Portlets and the storefront share a common JSESSIONID that maintains session authentication. This session id is not set unless the customer explicitly logs into the storefront.

Check that the URL used in the portlet preference looks exactly the same as the storefront URL.

Check to see if there are any error messages in the log related to content repository authentication. If there are, ensure that a user name and password is set. When in the Portal Administration console under repository, you can visibly see the admin user's login id (or a user with sufficient permission e.g. admin) and a masked password. If the field is blank, set the password by clicking on 'change password' and reenter your info.

Portlets application will need to be resynchronized or redeployed after any changes to settings.

Cannot checkout

Symptom:
Get error dialog box stating "Encryption keys must be greater than 24 characters"

Solution:
AquaLogic Commerce Services requires an encryption key used when storing credit card information for return customers (table TORDERPAYMENT). This key must be configured in commerce-config.xml. Look for the "encryption.key" element in the security node. For more information on the configurations in this file, please see the **commerce-config.xml** section above.

Symptom:
"404" when attempting to checkout or access "My Account"

Solution:

1. If you have disabled SSL, ensure that you have set the "forceHttps" property of the "authenticationProcessingFilterEntryPoint" bean to "false" in acegi.xml (both Commerce Manager and Storefront). For more information see the **acegi.xml** section above.
2. If SSL is enabled, ensure that the https port redirect has been configured correctly. In acegi.xml, look for the "portMappings" property in the "portMapper" bean. For more information, see the **acegi.xml** section above.
3. Application start-up is slow because a lot of time is spent building indexes.

Symptom:
In the logs, there are many INFO messages showing "Building Index"

Solution:
Index building is normal. Various indexes are built to speed up searches, for updating the rules engine (promotions), for calculating product recommendations, top sellers, etc. In a development environment where you are starting/stopping the application frequently, you may want to adjust these build times. In quartz.xml (one each for Storefront and Commerce Manager), each build job has a "start delay" and a "build interval". These values specify the number of seconds before the first build takes place after startup and the time delay between rebuilds respectively. You can edit these times, or you can comment out the jobs entirely in the "SchedulerFactory".

Please be advised that turning off the index building will have an effect on your application. For example, new products entered through the Commerce Manager will not be searchable in the Storefront if the search indexes are not rebuilt. Of course, turn the index building back on when you are ready to deploy to production.

Import jobs don't run

Symptoms:

1. In the "Preview Data" window, the following error appears:
   ERROR: Title name can only contain the following characters: digital, alphabetic, space and _: Column -1
2. The sample data displayed for row 0 appears to contain a mix of gibberish.

Solution:
Make sure that the character encoding of the import file is UTF-8, which is the default file encoding for data files. Alternatively, edit the commerce-config.xml file for the Commerce Manager Server, changing the "datafile.encoding" property from UTF-8 to the encoding of your choice. Be aware that changing this value will affect the encoding for import and export files, as well as reports files.

Unable to bring up WLP after successful install

Symptoms:

1. WLP is installed successfully on Solaris, and a user domain can be created via the configuration wizard.
2. Attempting to bring up the domain results in the WLP being forced to shut down with the following error message:

```
<Error> <Search> <BEA-415614> Unable to connect to the FullTextSearch engine instance
associated with repository Community_Repository in 36 attempts over a 180 second period, using
repository configuration settings of engine host localhost and index port 9001. FullTextSearch will
be not be available for this repository. Please ensure the repository configuration settings are
correct, and the FullTextSearch engine processes are running on this host/port. Check the
FullTextSearch engine logfiles for errors and warnings and verify adequate disk space exists.
```

Solution:
This problem is related to the BEA Autonomy content management search application that is a part of WLP. Autonomy is not supported on Solaris on x86 platforms. More information on the configurations supported by Autonomy can be found at http://edocs.bea.com/platform/suppconfigs/configs100/100_over/autonomy.html.

To work around this issue, you will need to edit the startWeblogic.sh file for WLP, changing the CONTENT_SEARCH_OPTION parameter from **FULL** to **none**:

```
if [ "${CONTENT_SEARCH_OPTION}" = "" ] ; then
        # If not set externally, this content domain will start the Full Autonomy engine stack.
        CONTENT_SEARCH_OPTION="none"
        export CONTENT_SEARCH_OPTION
fi
```

Make sure that the character encoding of the import file is UTF-8, which is the default file encoding for data files. Alternatively, edit the commerce-config.xml file for the Commerce Manager Server, changing the "datafile.encoding" property from UTF-8 to the encoding of your choice. Be aware that changing this value will affect the encoding for import and export files, as well as reports files.

Load Balancing and Clustering (optional)

This section explains how to setup a cluster of WebLogic servers with Apache HTTP Server as a load balancer and rsync for file synchronization.

General Clustering Architecture

A server cluster is a group of computers that act as one server. The most common use of a cluster is to increase performance and/or availability. When setup as a load balancing cluster, the work goes through one server and then is distributed among all nodes in the cluster.

Cluster Components

The following are the typical components of a load balancing cluster.

- Load Balancer – All traffic to your website goes through this server and it distributes the work to the nodes in the cluster.
- Cluster Nodes – The servers in the cluster that each have an instance of the web application and perform all of the work.
- File Synchronization Server – Synchronizes changes to common files across all servers in the cluster.

AquaLogic Commerce Services has been architected to run on a load balanced cluster. A typical configuration for this would be as follows.

- Load Balancer – An Apache server with a load balancing module configured.
- Cluster Nodes – Multiple application servers setup in a cluster each with the AquaLogic Commerce Services Storefront deployed to it.
- Commerce Manager Server – One application server with the AquaLogic Commerce Services Commerce Manager deployed to it.
- File Synchronization Server – An rsync server setup on the Commerce Manager Server and an rsync client setup on each Cluster Node.

Clustering Your Application Server

This section provides you with the tools and information you need to setup an application server cluster with WebLogic.

WebLogic 9.2 Clustering

This section describes how to setup a WebLogic 9.2 cluster on Linux servers using the BEA WebLogic Scripting Tool (WLST). WLST is a scripting tool installed with WebLogic that allows for command-line configuration of the WebLogic server. The set of scripts provided here along with these instructions will allow you to setup a WebLogic cluster in your environment with ease.

Clustering Scripts Setup

In preparation for running the WLST scripts to create a cluster, you will need to download the WLST Clustering Scripts package at http://edocs.bea.com/alcs/docs51/pdf/wlstClusteringScripts.zip (ZIP, 7KB) and unzip it to a directory. After that, configure the environment variables in all of the scripts to match your environment. A description of the variables that will need to be set can be found in the section Environment Variables to Configure in the WLST Scripts. After that, copy the files to the servers that you will be using in the cluster. You will then need to navigate to this directory on a server to execute any of the scripts on that server.

Steps to Create the WebLogic Domain and Setup the Cluster

On the Server hosting the AdminServer:

1	Create a domain with AdminServer and a cluster of two managed servers	WL_HOME/common/bin/wlst.sh createcluster.py DOMAIN_NAME CLUSTER_NAME
2	Start the node manager and AdminServer	WL_HOME/common/bin/wlst.sh startadminserver.py DOMAIN_NAME
3	Setup JDBC Data Source	WL_HOME/common/bin/wlst.sh createjdbc.py t3://AdminServerIP:AdminServerHttpPort CLUSTER_NAME
4	Pack the domain	WL_HOME/common/bin/pack.sh -managed=true -domain=DOMAIN_PATH -template=DOMAIN_TEMPLATE -template_name=DOMAIN_TEMPLATE_NAME

On all servers not hosting the AdminServer:

5	Unpack the domain	Copy DOMAIN_TEMPLATE from the server hosting the AdminServer to the same directory on all other servers that are apart of the cluster and then execute the following command on the servers to create the base domain: WL_HOME/common/bin/unpack.sh -domain=DOMAIN_PATH -template=DOMAIN_TEMPLATE
6	Enroll the node manager with AdminServer and then start it	Execute the following command on the servers to setup the node managers and start them: WL_HOME/common/bin/wlst.sh enrollnodemanager.py t3://AdminServerIP:AdminServerHttpPort DOMAIN_NAME

On the Server hosting the AdminServer:

7	Deploy the AquaLogic Commerce Services code to the cluster	WL_HOME/common/bin/wlst.sh deploy.py t3://AdminServerIP:AdminServerHttpPort DEPLOYMENT_NAME APPLICATION_PATH CLUSTER_NAME
8	Start the cluster	WL_HOME/common/bin/wlst.sh startcluster.py t3://AdminServerIP:AdminServerHttpPort CLUSTER_NAME

Additional Useful Scripts

To create another server in the cluster, execute this command after step 1	WL_HOME/common/bin/wlst.sh createmanagedserver.py DOMAIN_NAME CLUSTER_NAME
To shut down server ServerName	WL_HOME/common/bin/wlst.sh stopserver.py DOMAIN_NAME ServerName
To remove a deployment	WL_HOME/common/bin/wlst.sh undeploy.py

	t3://AdminServerIP:AdminServerHttpPort DEPLOYMENT_NAME

Descriptions of Constants Used in Setup Steps

BEA_HOME	The BEA Home directory where files common to all BEA products are stored (eg. /opt/bea/)
WL_HOME	The WebLogic Server product installation directory (eg. /opt/bea/weblogic92/)
AdminServerIP	The IP address of the server that the AdminServer is on
AdminServerHttpPort	The port for the AdminServer to listen to http requests on
DOMAIN_NAME	The name of the clustered domain to be created (eg. alcsclusterdomain)
DOMAIN_PATH	BEA_HOME/user_projects/domains/DOMAIN_NAME (the path to the clustered domain)
DOMAIN_TEMPLATE	The filename of the domain template to be created or accessed (eg. BEA_HOME/user_templates/alcsclusterdomain_managed.jar)
DOMAIN_TEMPLATE_NAME	The descriptive name of the domain template to be created (eg. "ALCS Clustered Domain")
CLUSTER_NAME	The name of the cluster to create (eg. wlsCluster)
DEPLOYMENT_NAME	The name for the deployment of AquaLogic Commerce Services application code (eg. alcssf_cluster_deployment)
APPLICATION_PATH	The path to where the AquaLogic Commerce Services application has been setup (eg. /home/build/alcs_weblogic/com.elasticpath.sf/)

Environment Variables to Configure in the WLST Scripts

BEA_HOME	The BEA Home directory where files common to all BEA products are stored (eg. /opt/bea/)
WL_HOME	The WebLogic Server product installation directory (eg. /opt/bea/weblogic92/)
JAVA_HOME	The root directory of the Java JDK install that is used to run WebLogic (eg. /opt/j2sdk)
AdminServerIP	The IP address of the server that the AdminServer is on
AdminServerHttpPort	The port for the AdminServer to listen to http requests on
AdminServerHttpsPort	The port for the AdminServer to listen to https requests on
AdminServerPassword	The password used to connect to the AdminServer as default user weblogic
Machine1IP	The IP address of the server hosting the first managed server in the cluster

Machine1Name	The machine name of the server hosting the first managed server in the cluster
Server1HttpPort	The port for the first managed server to listen to http requests on
Server1HttpsPort	The port for the first managed server to listen to https requests on
Server1Name	The name to use for the first managed server in the cluster (eg. alcsServer1)
Machine2IP	The IP address of the server hosting the second managed server in the cluster
Machine2Name	The machine name of the server hosting the second managed server in the cluster
Server2HttpPort	The port for the second managed server to listen to http requests on
Server2HttpsPort	The port for the second managed server to listen to https requests on
Server2Name	The name to use for the second managed server in the cluster (eg. alcsServer2)
JdbcName	The descriptive name of the JDBC data source (eg. ALCS)
JndiName	The JNDI name of the JDBC data source (eg. jdbc/alcsjndi)
Url	The JDBC connection URL (eg. jdbc:oracle:thin:@11.11.1.111:1111:alcs)
JdbcDriverName	The name of the JDBC driver (eg. oracle.jdbc.OracleDriver)
DbUserName	The username for accessing the database
DbUserPassword	The password for accessing the database

Useful Websites

Example setting up a clustered deployment on a single server with WLST	http://dev2dev.bea.com/pub/a/2006/03/wlst-clustered-deployment.html?page=1
Steps for Creating and starting a managed server on a remote machine	http://edocs.bea.com/common/docs92/pack/tasks.html#wp1068348
WLST Documentation	http://edocs.beasys.com/wls/docs92/config_scripting/
WLST Command and Variable Reference	http://e-docs.bea.com/wls/docs92/config_scripting/reference.html
WebLogic Server Mbean Reference	http://e-docs.bea.com/wls/docs92/wlsmbeanref/core/index.html

Load Balancing

Apache HTTP Server may be used along with an installed module to load balance among a cluster of application servers. This section explains how to setup Apache HTTP

Server and the sub-sections give the details on how to setup either the JK Connector module or the WebLogic Apache HTTP Server Plug-in module for load balancing. The JK Connector module can load balance a cluster of Tomcat or JBoss servers. WebLogic provides the WebLogic Apache HTTP Server Plug-in module to be used with Apache for load balancing WebLogic servers.

Setup of the Apache HTTP Server as a Load Balancer

- Download the latest version of Apache that can be used with the module that you will be using from the Apache Downloads Page at http://httpd.apache.org/download.cgi.

  If you download the source code and build it based on the instructions, don't forget to enable ssl when you do the configuration.

  ```
  $ {APACHE_HTTP_SERVER_SRC_DIR}/configure --enable-ssl
    --enable-so -enable-mods-shared="proxy \
    proxy_http proxy_ftp proxy_connect headers cache
    disk_cache mem_cache"
    --prefix={INSTALL_FULL_PATH}
  $ {APACHE_HTTP_SERVER_SRC_DIR}/make
  $ {APACHE_HTTP_SERVER_SRC_DIR}/make install
  ```

- Setup SSL for the Apache server
  - Create an SSL key and certificate
    Refer to the article at http://www.samspublishing.com/articles/article.asp?p=30115&amp;seqNum=4 for instructions on how to do this.
  - Copy the SSL key to APACHE_HOME/conf/, renaming the file to "server.key".
  - Copy the SSL certificate to APACHE_HOME/conf/, renaming it to "server.crt".
- Apache Http Server Control
  The following commands can be used to start and stop the Apache server. See the documention on the version of Apache that you downloaded for the full details of all commands.

```
• Start with ssl enabled:
• {APACHE_HOME}/bin/apachectl startssl
•
• Stop:
• {APACHE_HOME}/bin/apachectl stop
```

Configuring Apache with WebLogic Apache Http Server Plug-in

This section explains how to setup Apache HTTP Server with WebLogic's Apache Http Server Plug-in module to load balance a cluster of WebLogic 9.2 servers. See BEA's

WebLogic 9.2 Apache HTTP Server Plug-In documentation at http://e-docs.bea.com/wls/docs92/plugins/apache.html for more detailed instructions.

Load Balancing Setup with the WebLogic Apache HTTP Server Plug-in

1. Setup the Apache WebLogic clustering configuration files
    - Unzip the Apache WebLogic cluster configuration file at http://edocs.bea.com/alcs/docs51/pdf/apacheWebLogicClusterConfiguration.zip (ZIP, 2KB) to APACHE_HOME/conf/ on the Apache server.
    - Configure the values of the elements in the unzipped files weblogic.conf and ssl-weblogic.conf to match your environment. The following are descriptions of the elements in the files that will need to be configured:

APACHE_HOME	The BEA Home directory where files common to all BEA products are stored (eg. /opt/httpd.2.0.52/)
APACHE_IP	The IP address of the server hosting Apache
APACHE_MACHINE_NAME	The descriptive machine name of the server hosting Apache
APACHE_HTTP_PORT	The port that the Apache server will be listening to http requests on
APACHE_HTTPS_PORT	The port that the Apache server will be listening to https requests on
SERVER1_IP	The IP address of the server hosting the first managed server in the cluster
SERVER1_HTTP_PORT	The port that the first managed server is listening to http requests on
SERVER2_IP	The IP address of the server hosting the second managed server in the cluster
SERVER2_HTTP_PORT	The port that the second managed server is listening to http requests on
APPLICATION_CONTEXT_NAME	The name of the AquaLogic Commerce Services storefront context. Apache will only handle requests with this context in it. If the context is empty or you want Apache to handle all requests against APACHE_HTTP_PORT/APACHE_HTTPS_PORT, then this line can be taken out of the files.

2. Setup Apache Http Server Plug-in Module
    - Copy the correct version of the WebLogic Apache Http Server Plug-in for the Operating System of your Apache server from the WebLogic install to APACHE_HOME/modules. See the Apache HTTP Server Plug-In

documentation at http://e-docs.bea.com/wls/docs92/plugins/apache.html to determine which is the correct version to copy.

- Comment out the following line in APACHE_HOME/conf/httpd.conf so it can be replaced with a clustered WebLogic SSL configuration file.

```
# Include conf/ssl.conf
```

- Add the following lines to APACHE_HOME/conf/httpd.conf under the other LoadModule lines to include the weblogic clustering configurations (replacing mod_wl_20 with the correct name of the server plug-in module that you copied over).

```
LoadModule weblogic_module  modules/mod_wl_20.so
<IfModule mod_ssl.c>
   Include conf/ssl-weblogic.conf
</IfModule>
Include conf/weblogic.conf
```

File Synchronization with rsync

AquaLogic Commerce Services uses rsync to synchronize asset and template files between a Commerce Manager server and Storefront servers.

If your Commerce Manager and Storefront are located on the same server, you do not need to setup a rsync server.

If they are located on different servers, however, then you will need to setup a rsync daemon on the server hosting the Commerce Manager and rsync clients on all Storefront servers.

Download rsync

- Download the rsync software from the rsync download page at http://samba.anu.edu.au/rsync/download.html.
  The rsync server and client are bundled together. If you downloaded the source code, you can run the following commands to build it (a C compiler is required for this).

```
• {RSYNC_SRC_DIR}/configure
• {RSYNC_SRC_DIR}/make
• {RSYNC_SRC_DIR}/make install
```

Set Up the rsync Server

1. Setup the rsync daemon on the Commerce Manager server (replacing SF1_SERVER_IP and SF2_SERVER_IP with the IP addresses of the servers that you will be using as the nodes in the cluster and ALCS_INSTALL_PATH with the path to the root of the AquaLogic Commerce Services code base).

```
motd file = /etc/rsyncd.motd
```

```
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
hosts allow = SF1_SERVER_IP, SF2_SERVER_IP
exclude = .svn .svn/* *.bak

[PD:assets]
  path = ALCS_INSTALL_PATH/com.elasticpath.cm/template-resources/assets
  comment = assets

[PD:templates]
  path = ALCS_INSTALL_PATH/com.elasticpath.cm/WEB-INF/templates
  comment = view templates
```

2. Create the welcome message file /etc/rsyncd.motd.

```
Welcome to rsyncd server.
```

3. Start the rsync daemon.

```
rsync --verbose  --progress --stats --compress --rsh=/usr/local/bin/ssh \
    --recursive --times --perms --links --delete \
    -daemon
```

Setup the rsync Client on the Storefront Servers

1. Create a new file /sbin/rsync_alcs as follows (replacing CM_SERVER_IP with the IP address of the server that will be hosting the Commerce Manager and ALCS_INSTALL_PATH with the path to the root of the AquaLogic Commerce Services code base).

```
#!/bin/bash

rsync --verbose  --progress --stats --compress --recursive --times --perms
  --links --delete
  CM_SERVER_IP::assets/*  ALCS_INSTALL_PATH/com.elasticpath.sf/template-
resources/assets

rsync --verbose  --progress --stats --compress --recursive --times --perms
  --links --delete
  CM_SERVER_IP::templates/*  ALCS_INSTALL_PATH/com.elasticpath.sf/template-
resources/templates
```

1. Execute "chmod 744 /sbin/rsync_alcs"
2. Create a new cron job /etc/cron.d/rsync as follows:

```
1.# run rysnc every 10 minutes
2.*/10 * * * * root /sbin/rsync_alcs
```

⚠ Keep in mind that if a large number of files need to be replicated, it may take a while before they are transferred to each Storefront. If you are planning on importing a lot of products into a live production catalog, you should first upload the media files to the Commerce Manager application and let the replication finish before running the import.

References

Linux rsync tutorial: http://everythinglinux.org/rsync/