Configuration Guide

CONFIGURATION GUIDE
1 PURPOSE
2 CONFIGURATION BASICS
2.1 Editing Configuration Files 5
2.1.1 Using Comment Marks 5
2.1.2 Using CQDE (Authoring Configuration Only) 5
2.1.3 Using the CRX Explorer 6
2.1.4 Using WebDAV 8
2.1.5 Using the Package Tool 9
2.2 Restarting WSM 9
2.3 Bootstrap Configuration 11
2.4 New Installation vs. Update Installation 11
2.5 Finding out More about a Configuration File 11
2.6 Managing Configuration Files Centrally 12
3 SYSTEM BEHAVIOR
3.1 Setting a Virtual URL 13
3.2 Setting the Document Root 14
3.3 Setting the Initial Context 16
3.4 Team Development 17
3.5 Logging 19
4 AUTHENTICATION
4.1 Default Authentication 22
4.1.1 Configuring the Header Authentication Handler 22
4.1.2 Setting the Scope of the Header Authentication Handler 23
4.1.3 Using the Simple Authenticator 24
4.2 Sessions 25
4.3 Log-in Forms 26
4.3.1 Writing the Log-in Form 26
4.3.2 Enabling Sessions 26
4.3.3 Configuring the Parameter Authentication Handler 27
4.3.4 Enabling the Parameter Authentication Handler 28
4.3.5 Authentication 29
4.4 LDAP 29
4.4.1 Setup 29
4.4.2 Authentication 35
4.4.3 Synchronizing 37
4.4.4 Batch Synchronizing 37
4.4.5 Dynamic Synchronizing 44
4.4.6 Mixed Synchronizing 48
4.4.7 Managing Access Rights 48
4.5 Trusted Authentication 49
4.5.1 Configuring the Trusted Authentication Handler 49
4.5.2 Enabling the Trusted Authentication Handler 51
4.5.3 Enabling the Trusted Authenticator 53
4.5.4 Enabling Trusted LDAP Authentication 54
5 TOOLS
5.1 Link Checker 55
5.1.1 Internal Link Checker 55
5.1.2 External Link Checker 58
5.2 WSM JMX Console 58
6 PERFORMANCE
6.1 Memory 60
6.2 Caching 60
6.3 Search 63

7 SYSTEM ARCHITECTURE
7.1 High Availability 65
7.2 Hot Backup 65
8 WSM SERVLET ENGINE
8.1 Connecting 66
8.2 Adding a Web Application 67
8.3 Removing a Web Application 68
8.4 Starting and Stopping Web Applications 68
8.5 Changing the Administrator Password 69
8.6 Java Virtual Machine Information 70

1 Purpose

This section tells you how to configure WSM on your system. Note that WSM is pre-configured to work on most systems. You only need to configure it for advanced setups.

2 Configuration Basics

WSM uses configuration files for almost every aspect of its internal workings. Configuration files determine all file interaction, starting up, the structure of all internal files, external data connections, and much more.

Therefore, you can configure almost anything in WSM. Most settings are internal, and it is best to keep them as they are. In most environments, WSM runs out-of-the box, and you generally do not have to configure it for use in non-productive environments.

This guide aims to point out the settings you can change and how to best configure WSM.

2.1 Editing Configuration Files

You can edit the configuration files in the CRX repository of the authoring or publishing environment.

2.1.1 Using Comment Marks

In all configuration files, you can use the comment mark to add comments. If you put a configuration setting in comment marks, it is not used.

For clarity, this guide shows only the active configuration settings and omits any settings in comment marks. Often, you can add a setting by removing the comment marks. Alternately, you can remove a setting by placing it in comment marks.

If you use comment marks for adding and removing configuration settings, you have a better overview of the available options, and the changes you have made.

2.1.2 Using CQDE (Authoring Configuration Only)

You can edit a configuration file on the authoring environment using CQDE (the WSM development environment). Because CQDE is usually not installed on a publishing instance, you cannot use it to modify configuration files there.

To change a configuration file for the authoring environment, proceed as follows:

1. Open CQDE. Login to the WSM authoring environment (at the path /author/cqde).
2. Click the **Resources** tab.
3. Open the **/config** folder. Right-click the file you want to edit, and then click **Check Out**.

4. Double-click a configuration file to edit it. When asked, click **Yes** to check out the file.

5. After you have modified the file, click **Save** to save the changed.

6. After you have saved the file, right-click it, and then click **Check In**. Click **OK** to check in the file.

Note: If you do not check in the file, WSM may continue to use the old, checked-in version of the configuration file, and your changes have no effect.

2.1.3 Using the CRX Explorer

All files in the ContentBus are stored as nodes in CRX. You can use the CRX explorer to edit the settings. You can use this if WSM is unavailable, for example, because of a configuration issue.

Because CRX stores WSM nodes as binary properties, you cannot edit the text of the configuration files using the CRX Content Explorer. You must export the file to the file system and re-import it.

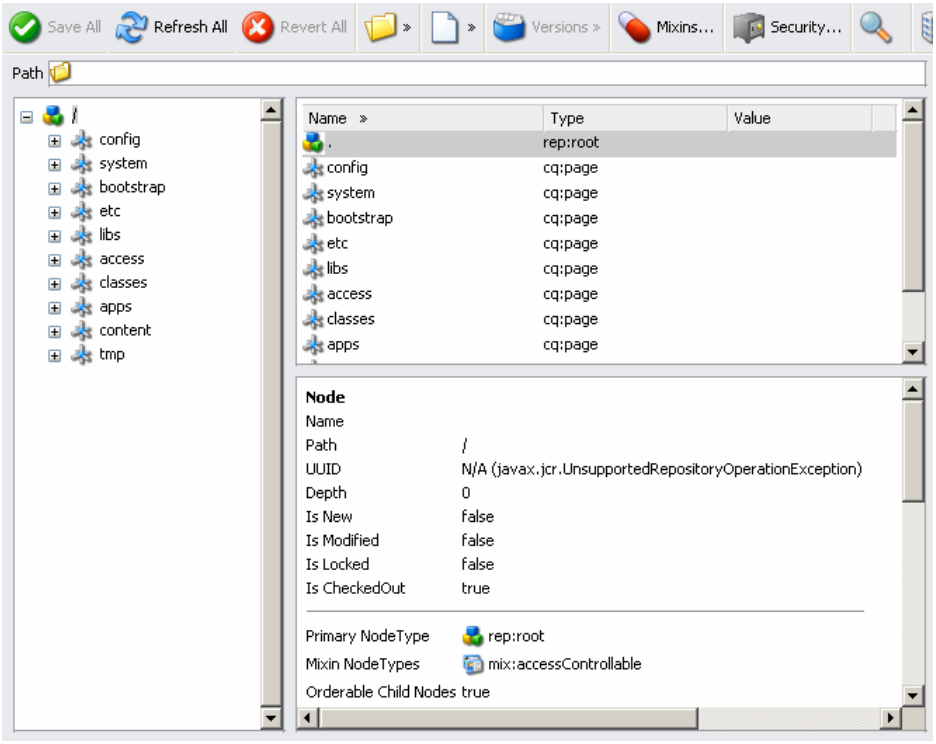
To start the CRX Content Explorer, proceed as follows:

1. Login to CRX. For example, type **localhost:4402/crxauthor** in the address bar of your Web browser. The CRX console appears.

2. Click **Log In**. Type your user name and password, for example, **admin** and **admin**.

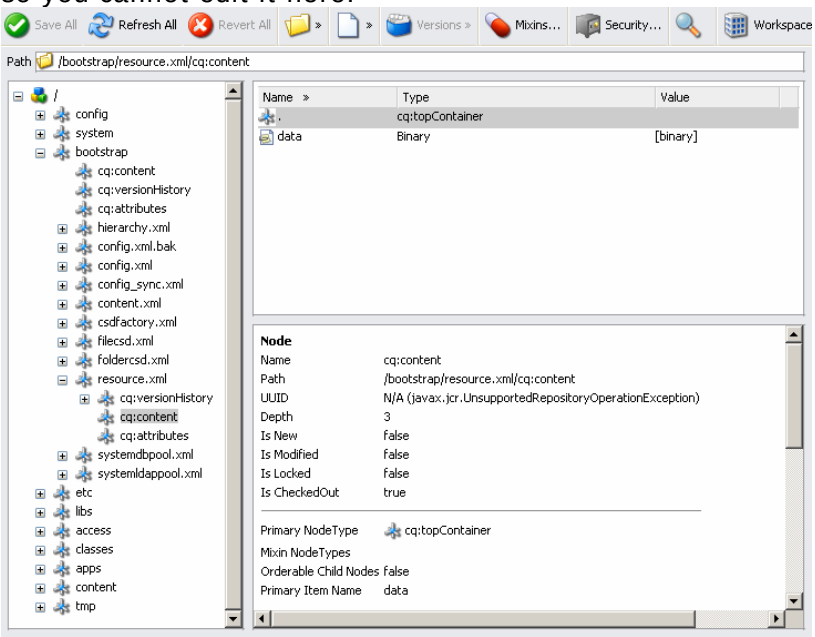
3. Click **Content Explorer**. The Content Explorer opens in a new window.

The Content Explorer displays WSM's repository structure. The configuration files are in the /bootstrap and /config folders.



To export a configuration file to the file system, proceed as follows:

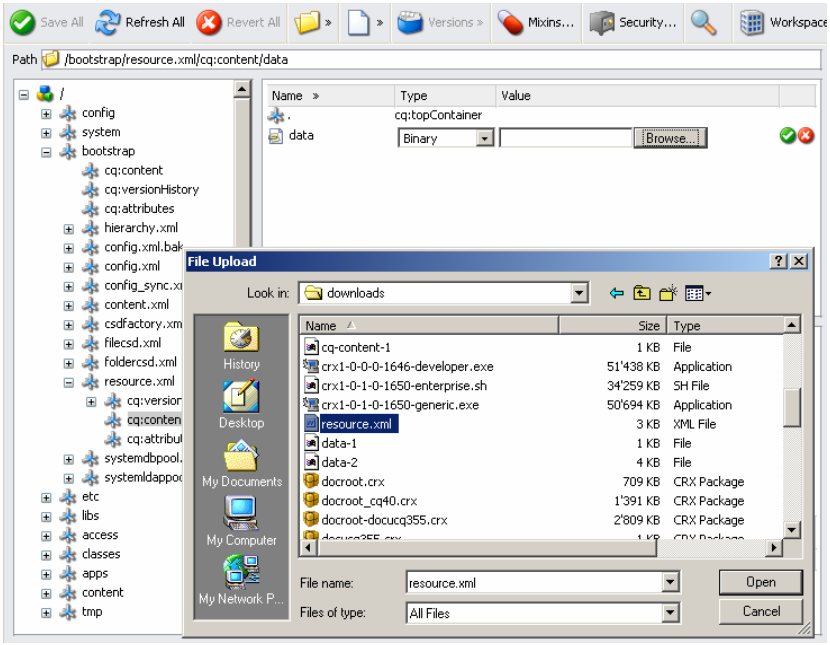
1. Unfold the configuration file you want to modify by clicking the plus sign next to it. Click the **cq:content** node to display the content in the top right pane. The content node has a property named data. Its content is stored in binary format, so you cannot edit it here.



2. Right-click **[binary]**, and then click **Save Link As**. Type the name of the configuration file (for example “resource.xml”, and then click **Save**.

You can now edit the saved file in the file system. To upload the file into WSM again, proceed as follows:

1. Double-click **data**. The data property changes to a text field and a file select field.
2. Click **Browse**. A file select window opens. Double-click the configuration file (for example “resource.xml”) to upload the file.



3. Click **accept** (the green check icon) to accept the change.
4. Click **Save All** (top left) to save the change.

2.1.4 Using WebDAV

Using WebDAV, you can access all configuration files on both the authoring and the publishing environment. To access the environment from a Windows computer, you need a WebDAV connection to WSM. Proceed as follows:

1. In the folder **My Network Places**, click **Add Network Place**. The Add Network Place Wizard opens.
2. Type the location of the WebDAV network place. For example, to access a local publishing environment, type **http://localhost:4402/publish/etc/webdav/resource.dav**.
3. Log in with your WSM user name and password.
4. Type a name for the connection, such as **Publishing Environment**.
5. Click **Finish** to create the new network place.

With the WebDAV folder, you can now edit the configuration files in the ContentBus. Note that depending on the WebDAV client you use, you can edit the files directly in the WebDAV folder or you may have to copy them to your computer, edit them, and copy them back.

2.1.5 Using the Package Tool

If you modify few settings in few configuration files, it is usually faster and safer to modify the files on all the environments by hand. If you need to modify several files, or if you modify the files repeatedly, you can use the WSM package tool to publish the modified files to other WSM systems.

For details on how to use the package tool, see the WSM User Guide. If you repeatedly need to publish different configuration files to different WSM systems, see Managing Configuration Files Centrally.

Note: If you use the package tool to publish configuration files, make sure that the files do not contain settings that are particular to the WSM system they originate from, such as the port number.

2.2 Restarting WSM

If you change a configuration file, the change may take effect as follows:

- When you save the configuration file.
- When you check in the configuration file.
- When you restart WSM.

When a change becomes effective depends on how fundamental the change is. In some cases, WSM may display an error message if you modify several configuration files and one change becomes effective immediately, while a related change is not yet effective.

Restarting one WSM Instance

If you have modified the configuration of a WSM instance, you have to restart it (not all changes require this). To restart a WSM instance, proceed as follows:

1. In the address bar of your Web browser, type the path to the WSM Servlet Engine Administration Screen, for example http://localhost:4402/admin
2. Login with the servlet engine user name and password. These are different from the WSM passwords. The default is **admin** and **admin**.

3. If you have changed the configuration of the authoring environment, click Stop, and then click Start next to the Author instance.

4. If you have changed the configuration of the publishing environment, click Stop, and then click Start next to the Publish instance.

WSM now restarts with the new configuration.

Restarting the WSM Servlet Engine

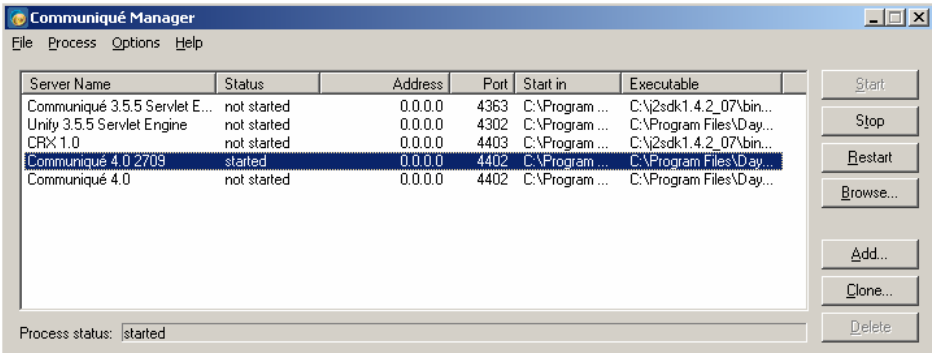
If you restart the WSM Servlet Engine, you automatically restart the WSM administration servlet, all WSM instances (typically authoring and publishing), and all servlets that run in the engine.

This is usually not necessary. Restart the Servlet Engine if:

- A WSM instance does not start and you cannot locate a specific problem.
- You have changed the Servlet Engine configuration.
- You have modified the Servlet Engine, for example, by adding program libraries.

To restart the Servlet Engine on a Windows computer, proceed as follows:

1. Double-click the Servlet Engine icon in the task bar. The WSM Manager opens.
2. Click the WSM server you want to restart, and then click **Restart**.



2.3 Bootstrap Configuration

WSM uses the bootstrap configuration to start the ContentBus. After the ContentBus starts, WSM reads the rest of the configuration files from the /config folder in the ContentBus.

The bootstrap configuration is available at three locations:

- In the ContentBus in the /bootstrap folder. Edit the files here.
- In the CRX repository /bootstrap folder. Edit these files if you cannot start the ContentBus.
- In the file system in the /server/runtime folder. Do not change these files.

2.4 New Installation vs. Update Installation

If you have a new WSM 4 installation, WSM stores the content in the CRX repository. If you use an update installation from a previous WSM version, WSM 4 uses the ContentBus repository.

To configure the repository of a **new installation**, use the files in the configuration folder **/repository**. You do not need to modify any files in the /contentbus folder.

To configure the repository of an **update installation**, use the files in the configuration folder **/contentbus**.

Note that some files are the same in both folder, and some are specific to CRX or the ContentBus. In some cases, files were renamed, or configuration settings were moved to a different file.

2.5 Finding out More about a Configuration File

Most configuration files are XML files based on a DTD (Document Type Definition). WSM stores the DTD in the ContentBus. If you want to know which options are available for a configuration file, read the DTD. Usually, either the configuration file or the DTD is fully documented.

Note: In some cases, the DTD is not available in the ContentBus. In these cases, the DTD is in the JAR file cq3.jar or cq4.jar in the folder data\author\bin\lib or data\publish\bin\lib in the WSM installation directory.

2.6 Managing Configuration Files Centrally

If you develop, test, and publish on different Web sites, you may typically have four configurations:

- The development site authoring configuration
- The test site authoring configuration
- The live site authoring configuration
- The live site publishing configuration

Ideally, these configurations are as similar as possible. If the configurations are different, for example because they run on computers with different port numbers, you can use **instance mapping** to manage all configuration files on a central location, such as the development authoring site. Note that this works only if you use the package tool to publish the files.

Instance mapping allows you to rename files in a package depending on the name of the target system. For example, to manage the configuration files for the above instances centrally, proceed as follows:

1. Create a default file in the development site, for example, named myConfig.xml.
2. On the same site, create the file myConfig_test_author.xml with the test site configuration.
3. Create the files myConfig_live_author.xml and myConfig_live_publish with the live site configuration.
4. In the package tool, create a mapping for each of the three target sites, which changes the name of the configuration file to myConfig.xml.

Now when you create the packages for each site, the configuration files are automatically exchanged.

Note: You can specify the mapping only on a per-file basis. If you have four configuration files and three target sites, you have to specify 12 mappings.

3 System Behavior

These settings allow you to change the default behavior of WSM.

3.1 Setting a Virtual URL

The URL mapping allows you to convert virtual (that is, nonexistent) URLs to Web site URLs on-the-fly. It is typically used in the following situations:

- You want to use simple start URLs. For example, when a user types “www.mycompany.com”, the user arrives at the page “www.mycompany.com/en/default.html”.
- You have moved a page and want to redirect users who try the old address.
- You use tools that expect URLs that your Web site does not have.

Default

By default, WSM uses a few internal redirect requests, for example to start the development environment.

The mapper.xml File

You can specify the conversion in the <fakeurls> section of the file config/delivery/mapper.xml in the ContentBus. It has the following structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mapper SYSTEM
"cq:/system/resources/dtd/delivery/mapper.dtd">

<mapper>
  <fakeurls>
    <redirect from="/old" to="/new" />
    <redirect from="/simpleUrl" to="/complexUrl" />
  </fakeurls>

  <mapping direct="true">
    <map from="/content/" to="/" />
   <map from="/apps/*/docroot/" to="/" />
    <map from="/libs/*/docroot/" to="/" />
    <map from="/system/docroot/" to="/" />
  </mapping>
</mapper>
```

Adding a New Virtual URL

To add a new redirect, proceed as follows:

1. Open the configuration file config/delivery/mapper.xml.
2. Add a new redirect entry. In the “from” attribute, type the URL you want to redirect. In the “to” attribute, type the URL of the target you want to redirect to.

An example section can look as follows:

```
  <fakeurls>
    <redirect from="/en/juneOffer.html"
to="/en/julyOffer.html" />
    <redirect from="/myProduct.html"
to="/en/products/myProduct.html" />
<redirect from="/private" to="/en/private/login.html" />
    <redirect from="/ " to="/en/home" />
  </fakeurls>
```

Note: Entries must match perfectly with the request URL. Partial matches are ignored.

3.2 Setting the Document Root

WSM uses a “docroot” folder for files that it serves directly, without further internal processing. Such files are, for example, the images used for the Web site layout.

We recommend that you store all your files in the following places:

- Use the Media Library for files that authors upload, edit and use.
- Use the /docroot folder of your application for general files, such as the images you use in your layout. For example, you can store the company logo at the location “/apps/myApplication/docroot/logo.gif”

If you store files in these locations, you do not have to set the docroot. If you use other file locations, add the docroot as follows.

Default

By default, WSM checks the following locations for files:

- The location as it is presented in the URL
- The /content folder
- All /docroot folders in the /apps and /libs folder.
- The folder /system/docroot

The mapper.xml File

You can specify your docroot in the <mapping> section of the file config/delivery/mapper.xml in the ContentBus. It has the following structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mapper SYSTEM
"cq:/system/resources/dtd/delivery/mapper.dtd">

<mapper>
  <fakeurls>
  </fakeurls>

  <mapping direct="true">
    <map from="/content/" to="/" />
   <map from="/apps/*/docroot/" to="/" />
    <map from="/libs/*/docroot/" to="/" />
    <map from="/system/docroot/" to="/" />
  </mapping>
</mapper>
```

Each <map> element defines a mapping from an internal ContentBus handle to an external handle. The external handle is always set to “/”, meaning that the files in the docroot are available in the server root path.

WSM evaluates the entries from top to bottom. For example, if you type “www.myCompany.com/myFile.jpg”, WSM does the following:

- Because the “direct” attribute is set to “true”, it tries to locate the file in the ContentBus root. This fails.
- It tries to locate the file “/content/myFile.jpg”. If the file exists, it returns the file, if not, it continues.
- It looks for the file in all “/docroot” folders in the “/apps” folder.
- It looks for the file in all “/docroot” folders in the “/libs” folder.
- If the file is not found, WSM returns an error.

Adding Your Own Document Root

If you place your files in the /docroot folder of your application, or in the media library, you do not have to change the configuration. If you use a different folder to store the files, such as “apps/myApplication/files” add a <map> element to the <mapping> section as follows:

```
  <mapping direct="true">
    <map from="/apps/myApplication/files" to="/" />
    <map from="/content/" to="/" />
```

```
   <map from="/apps/*/docroot/" to="/" />
     <map from="/libs/*/docroot/" to="/" />
     <map from="/system/docroot/" to="/" />
   </mapping>
```

Using Sub Folders

You can use sub folders in your /docroot folder. For example, WSM maps the file

```
apps/myApplication/docroot/layout/white.gif
```

to

```
www.myCompany.com/layout/white.gif.
```

You do not have to change the configuration for this. Do not use folder names in the /docroot folder that exist on the Web site. This may lead to conflicts between docroot files and Web pages.

3.3 Setting the Initial Context

WSM needs a so-called “initial context” to manage external resources in the resource pool. If WSM runs on an application server, you can use the application server’s default initial context.

Creating a Default Initial Context

WSM needs a default initial context to use external resources in the resource pool. The contexts are stored in the file **resource.xml**, in the folder /bootstrap. To create the default initial context, add an empty <defaultinitialcontext> element as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE resource SYSTEM
"cq:/system/resources/dtd/resource.dtd">

<resource>

  <!-- Define the JNDI initial contexts -->
  <jndicontexts>
    <defaultinitialcontext>
    </defaultinitialcontext>
  </jndicontexts>

  <!-- Define resource pools -->
  <resourcepools>
  ...
  </resourcepools>

</resource>
```

Note: If WSM runs on an application server, you do not need to create a default initial context.

Adding an Initial Context

If you want to use resources from other contexts, for example from an application server other than the one WSM is running on, you can add other initial contexts. You can use these contexts when you specify external resources in the resource pool.

The contexts are stored in the file **resource.xml**, in the folder /bootstrap. To add a context, add an <initialcontext> element. The following example adds an initial context for a Weblogic application server:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE resource SYSTEM
"cq:/system/resources/dtd/resource.dtd">

<resource>
   <jndicontexts>
      <defaultinitialcontext>
      </defaultinitialcontext>

      <initialcontext name="weblogic">
         <param
            name="java.naming.factory.initial"
            value="weblogic.jndi.WLInitialContextFactory"
         />
         <param
            name="java.naming.provider.url"
            value="t3://localhost:7001"
         />
      </initialcontext>
   </jndicontexts>

   <resourcepools>
   </resourcepools>

</resource>
```

3.4 Team Development

If you use this WSM installation for development, and more than one developer works on it, we recommend using a separate class loader for each developer. Each class loader loads a separate copy of the Java classes, so if developer changes a Java class, the other users continue to use their version of the class.

Note: If several developers work on the same WSM installation and share the same class loader, each time a developer changes a class; the change immediately affects all other developers. Every time a developer makes a mistake at a critical location, WSM may stop working for all others as well.

Default Settings

By default, WSM uses a separate class loader for the super user and for each member of the user group “developer”.

The scripting.xml File

WSM stores the class loader preferences in the file scripting.xml, which looks as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE scripting SYSTEM
"cq:/system/resources/dtd/delivery/scripting.dtd">

<scripting>
  <scripthandlers>
  ...
  </scripthandlers>

  <errorhandler>
  ...
  </errorhandler>

  <classloader force="false">
      <user handle="/access/users/superuser" />
      <group handle="/access/groups/developer" />
  </classloader>

   <classpath>
  ...
   </classpath>

</scripting>
```

Using a Separate Class Loader for Each User

If the WSM installation is only used for development, you can use one class loader for each user. To do so, modify the <classloader> element as follows:

```
  <classloader force="true">
      <user handle="/access/users/superuser" />
      <group handle="/access/groups/developer" />
  </classloader>
```

Note: If you set the force attribute to “true”, WSM ignores the other settings for the class loader.

Using a Separate Class Loader for a User

To use a separate class loader for a user, add a user element as follows:

```
<classloader force="false">
    <user handle="/access/users/superuser" />
    <user handle="/access/users/myUser" />
    <group handle="/access/groups/developer" />
</classloader>
```

Using a Separate Class Loader for all Users of a Group

To use a separate class loader for each user of a group, add a group element as follows:

```
<classloader force="false">
    <user handle="/access/users/superuser" />
    <group handle="/access/groups/developer" />
    <group handle="/access/groups/myGroup" />
</classloader>
```

3.5 Logging

You can configure which types of log messages WSM logs, and where it writes the log files.

Default Settings

By default, WSM keeps the following logs for both the authoring and the publishing instance in the folder **/data/author/logs** and **/data/publish/logs**, respectively:

access.log	Logs all requests. This is a detailed list of who wanted what from WSM. You can configure this log in the file logspec.xml.
request.log	Logs all requests, response times and response status codes. Use this log to find out how fast and how correctly WSM answers requests. You can configure this log in the file logspec.xml.
error.log	Logs WSM's internal messages. By default, it logs errors, warnings and information messages. You can configure this log in the file logspec.xml.
crx-login.log	Logs all users who log in to the CRX repository.
crx-error.log	Logs the CRX messages for the WSM instance.
inboxnotifstart.log	The log file of the inbox notification.
newsletter.log	The log file of the newsletter library. If you do not use the library, you can ignore this

	file.
/replication	The replication folder contains one log per replication agent. Use these if you have problems with replication.

Note: Do not confuse these logs with the Web server logs or the servlet engine logs. In a default installation, the WSM Servlet Engine keeps its logs in the folder server/logs.

Configuring the WSM Logs

The file logspec.xml defines the files that WSM uses to log system information. Using the configuration file, you can specify different log files, restrict the maximum log file size, or change the output format for a log message.

The log files specified in this configuration file are created using the log4j framework. Refer to the log4j documentation for information on how to modify these settings.

You may want to configure different logs for development, authoring and publishing environments. To do this, configure the files separately on each WSM environment.

Configuring the Servlet Engine Log

The Servlet Engine keeps three log files in the folder server/logs:

- The file startup.log logs messages while the Servlet Engine starts. It is usually small, and you cannot configure it.
- The file server.log logs server messages such as the start of a Web application or Java errors. You can configure it in the file /server/etc/server.xml.
- The file access.log logs all server access information, such as page requests. You can partly configure it in the file /server/etc/server.xml.

Note: By default, the file access.log has the combined information of the access logs of the authoring and publishing instances. You cannot switch this log file off, but you can minimize its file size.

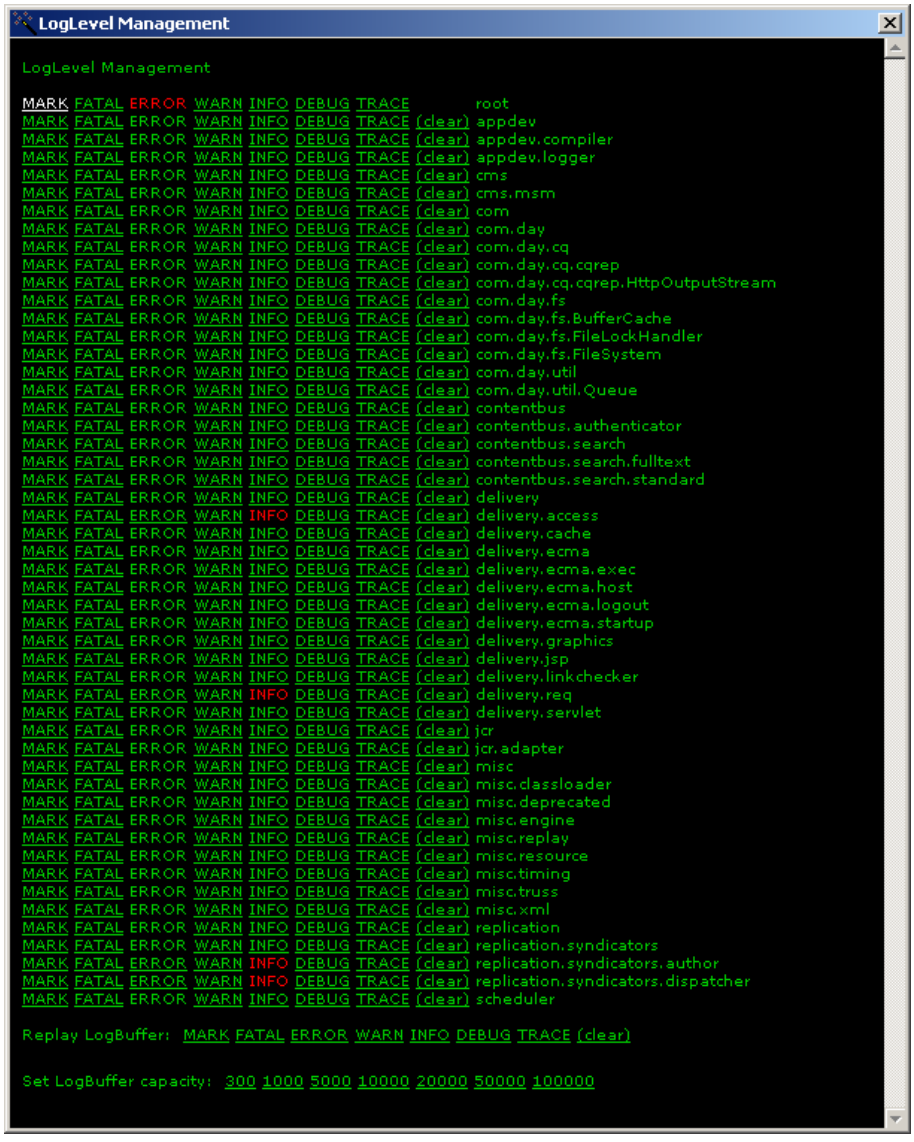
Changing the Log Level

The log level determines the amount of information that WSM writes into the log. For example, a log level of ERROR logs error messages, while a log level of INFO logs information messages. Log

levels include lower levels, so a level of INFO also logs warnings (WARN), errors (ERROR) and so on.

You can use the WSM Development Environment (CQDE) to change the log level of different parts of WSM. Proceed as follows:

1. Open the WSM Development Environment (CQDE) and connect to WSM.
2. In the Tools menu, click **Log Management**. The Log Level Management window opens.
3. Click the log level you want to use for each branch of WSM.



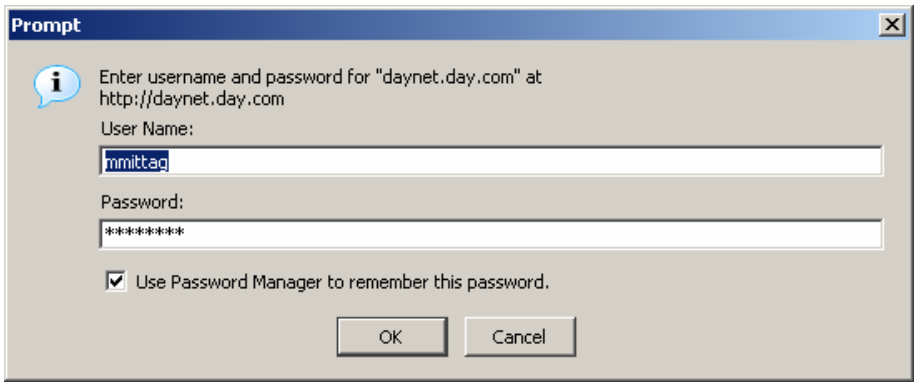
4 Authentication

Authentication is a three-step process:

1. An **authentication handler** reads the authentication information.
2. An **authenticator** evaluates whether the authentication information is sufficient. It returns the WSM user page of the user if successful, nothing ("null") if it did not find the user, or it throws a login exception if the user exists but authentication fails.
3. If authentication succeeds, WSM returns the requested page.

You can configure all of these steps separately. Note that WSM does not support all combinations in all cases.

4.1 Default Authentication



By default, WSM uses the header authentication handler to acquire login information, and it uses the simple authenticator to verify it. This means that users can log in to WSM using the browser's log-in window. WSM then tries to find a WSM user that matches the user name and password. If there is one, log-in is successful.

4.1.1 Configuring the Header Authentication Handler

The header authentication handler uses the browser's authentication mechanism. If a user requests a protected page, the browser opens a log-in window and requests the user name and password. After the user types these, the browser sends them to WSM for authentication.

Note: You cannot modify the appearance of the log-in window or the way the Web browser handles the details of the authentication process.

How the Header Authentication Handler Works

In a header authentication handler setup, a user can access a protected page as follows:

1. The user tries to access the protected page.
2. WSM returns a status code of “401” (unauthorized), and requests authentication using the “HTTP basic” authentication scheme.
3. The Web browser displays a log-in window, where the user types the user name and password. The browser submits this information to WSM in the HTTP header.
4. WSM checks if a user with the supplied user name and password exists and may see the page. If so, it returns the requested page.
5. On subsequent requests, the Web browser typically resubmits the authentication information, so the user does not have to log in again. This behavior is automatic and determined entirely by the Web browser.

Note: When you test your authentication scheme, keep in mind that the browser re-submits the login information automatically. If you make changes to the Web site, you may have to restart your Web browser to keep it from re-submitting outdated login information.

4.1.2 Setting the Scope of the Header Authentication Handler

The file auth.xml in the folder /delivery contains the list of packages. Each package specifies an authentication handler for a branch of WSM. By default, WSM uses the header authentication handler for the entire Web site, as follows:

```
<packages defaulthandler="header">
  <package default="none"
           handler="header"
           param="WSM"
           session="true">
    <include glob="*" />
    <exclude glob="/favicon.ico" />
   </package>
</packages>
```

Security

The header authentication handler uses no encryption. The user name and password are sent as uuencoded string. This means that the user name and password are not sent in plain text, but anyone can decode them.

4.1.3 Using the Simple Authenticator

The simple authenticator tries to find a WSM user with the user name and password that the authentication handler has acquired.

The simpleauthenticate.xml File

The simpleauthenticate.xml file in the /authenticators folder of the /repository folder specifies the behavior of the simple authenticator.

Note: If you have migrated from WSM 3.x, use the file in the /contentbus folder.

Usually, you do not have to modify the settings.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE simpleauthenticate SYSTEM
"cq:/system/resources/dtd/contentbus/authenticators/simpl
eauthenticate.dtd">

<simpleauthenticate>
    <trustedcredentials allow="true" />
    <emptypasswords allow="false" />
    <encryption encoding="utf-8" fallback="false" />
</simpleauthenticate>
```

The default settings are as follows:

- The simple authenticator works with trusted credentials.
- Empty passwords are not allowed.
- Passwords are encoded in UTF-8 Unicode.

Note: Do not allow empty passwords when you use LDAP. The LDAP authenticator can copy user information from the LDAP server to WSM. The users it creates have a valid user name, but no password. If you allow empty passwords, and you delete a user from the LDAP server, users may still log in using the copied WSM user and an empty password.

Using the Indexless Simple Authenticator

The simple authenticator uses WSM's internal search index to locate the user. If you have few users and high overall traffic, you can relieve WSM's internal search by using the indexless simple authenticator.

If you use a ContentBus installation, you can change the authenticator in the /repository folder, in the file securityservice.xml.

Note: If you have migrated from WSM 3.x, the settings are part of the file contentbus.xml in the /contentbus folder.

In the configuration file, modify the <authenticators> element as follows:

```
<authenticators defaultpackage="com.day.cq.contentbus">
    <authenticator class="IndexlessSimpleAuthenticator"
config="simpleauthenticate.xml" />
</authenticators>
```

The configuration file is the same for the standard simple authenticator and the indexless simple authenticator.

Note: If your Web site has many users, the indexless simple authenticator is much slower that the standard simple authenticator. Use it only if you have few users. Do not use it in authoring environments. Remove all unnecessary users from the site before you switch on the indexless simple authenticator.

4.2 Sessions

WSM can use sessions to store the login information and other user information. When a user a page from WSM, and you use session, the following happens:

1. The user requests the first page.
2. If the user does not have a session, WSM creates a session with a session ID.
3. If the user has logged in, WSM stores the login information in the session.
4. WSM returns the requested page and creates a session cookie on the user's computer. The cookie contains the session ID.
5. When the user requests the next page, WSM reads the session cookie to identify the user and load the session data.

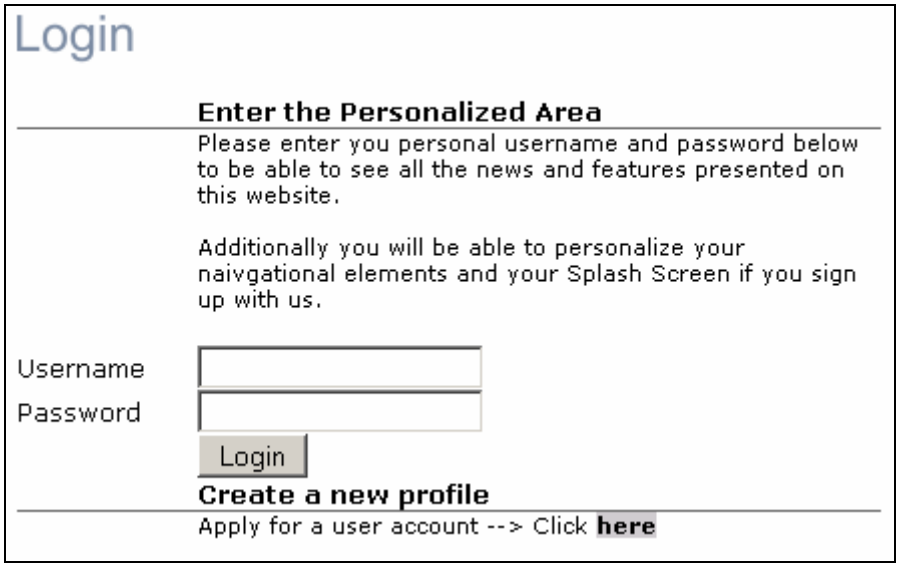
You can set the session cookie settings in the file **auth.xml** in the folder config/delivery. The cookie definition is in the <session> element. By default, it is commented out. To use sessions, remove the comment marks ("<!—" and "-->"), as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE auth SYSTEM
"cq:/system/resources/dtd/delivery/auth.dtd">

<auth>
  <session cookie="cq3session" ttl="-1"
persistent="persistent" />

...

</auth>
```

The <session> element has the following attributes:

- **cookie**: The name of the cookie.
- **ttl**: The time-to-live, which is the time in seconds that the cookie remains valid after the last request. Use “0” or “-1” to specify that the cookie always remains valid.
- **persistent**: Use **true** to save the cookie, so it is still available when the user re-starts the Web browser. Use any other value to delete the cookie when the user closes the Web browser.

4.3 Log-in Forms



A log-in form allows you to create a custom log-in screen and offers more control over all aspects of authentication.

4.3.1 Writing the Log-in Form

As a log-in form, you can use a standard HTML form that has a field for the user name and one for the password. The form can use the GET or the POST method. An example form is as follows:

```
<form method="POST" action="">
    <input name="userid">
    <input name="password">
    <input type="submit" value="Log In">
</form>
```

You can use any name for the user name and password field, but you have to configure the parameter authentication handler to use the same names. The above values are the default values for the parameter authenticator.

4.3.2 Enabling Sessions

The parameter authentication handler uses sessions. To use it, you need to enable sessions by removing the comment marks (“<!--“

and “-->” from the <session> element in the file **auth.xml** in the folder /config/delivery:

```
<auth>
  <session cookie="cq3session" ttl="-1"
persistent="persistent" />

...
```

4.3.3 Configuring the Parameter Authentication Handler

The parameter authentication handler uses form data for authentication. This allows you to create your own log-in screen that features the same layout as your Web site, and can offer additional functions, for example to register as a new user, or to retrieve a lost password.

Note: The parameter authentication handler requires sessions to work correctly. If you use it, do enable sessions.

How the Parameter Authentication Handler Works

A user can access protected content using the parameter authentication handler as follows:

1. The user tries to access a protected page.
2. WSM redirects the user to the login screen.
3. The user types the user name and password into a form and submits the form to WSM.
4. WSM checks if a user with the supplied user name and password exists and may see the page. If so, it returns the requested page. Along with the page, it returns a session cookie that contains the access information.
5. On subsequent requests, WSM uses the cookie to verify the user’s access rights, so the user does not have to log in again.

Security

The parameter authentication handler uses no encryption of its own. If you use it with unencrypted pages, it sends the form data as plain text. If you use encrypted pages (https), this authentication method is secure.

The parameterauth.xml File

In the /config/delivery folder, the **parameterauth.xml** file configures parameter authentication. You can set three values:

userid	The name of the form field that stores the user name.
password	The name of the form field that stores the password.
form	The URL of the form. The authenticator goes to this page if the user is not logged in, or if the log in fails.

```
<!DOCTYPE parameterauth SYSTEM
"cq:/system/resources/dtd/delivery/authenticators/paramet
erauth.dtd">

<parameterauth
    userid   = "userid"
    password = "password"
    form     = "/libs/Authentication/content/login.html"
/>
```

4.3.4 Enabling the Parameter Authentication Handler

Some of WSM's tools cannot use sessions. To use the parameter authentication handler, you have to define that these tools use the WSM default authentication handler, and that the rest of WSM (which includes the Web site content) uses the parameter authentication handler.

Excluding Incompatible Areas

Because several areas of WSM do not understand parameter authentication or sessions, you have to define that these areas use the standard header authentication handler.

In the file **auth.xml** in the folder /config/delivery, create one package for each area that cannot use parameter authentication. Note that you also have to switch off sessions for these areas, because WSM uses them by default if they are enabled. In the package list, remove the current package and specify the following packages (they are already in the file, but commented out):

```
<!-- WSM Development Environment -->
     <package default="none" handler="header" param="WSM
Development Environment" session="false">
         <include glob="/cqde" />
       <include glob="/system/cqde" />
       <include glob="/system/cqjd" />
       <include glob="/libs/TemplateWizard" />
       <include glob="/libs/CFC/content/wizards" />
       <exclude
glob="/libs/CFC/content/wizards/dialog/convertCFCDlg.html
" />
       <include glob="/libs/CFC/content/post.*" />
      </package>
```

```
<!-- WebDAV -->
     <package default="none" handler="header" param="WSM
WebDAV">
         <include glob="/etc/webdav" />
       </package>

<!-- Replication -->
     <package default="none" handler="header" param="WSM
Replication" session="false">
        <include glob="/system/replication" />
       </package>

<!-- System installation -->
     <package default="none" handler="header" param="WSM
Installer" session="false">
         <include glob="/system/installer" />
       </package>
```

Using the Parameter Authentication Handler

Below the packages that use the header authentication handler, add one package that specifies the parameter authentication handler for everything. Because WSM evaluates the packages from top to bottom, this package is used only for the areas that do not match a previous package.

```
     <package default="none" handler="parameter">
         <include glob="*" />
       <exclude glob="/favicon.ico" />
     </package>
```

Note that the icon favicon.ico is excluded. This is the icon that a Web browser may use for bookmarks to the Web site.

4.3.5 Authentication

For authentication, you can use either the simple authentication, which is the default for WSM, or the LDAP authentication. If you work in a trusted environment, you do not need log-in forms.

4.4 LDAP

WSM can interact with a central LDAP server that stores user information. WSM can use the central server to verify login information, or to create users and groups on-the-fly, if a user that is stored on the LDAP server logs in to WSM.

4.4.1 Setup

Because the LDAP user authentication changes several core aspects of how WSM handles authentication and authorization, you need to edit a variety of configuration files.

You can use the LDAP authentication with the default authentication, the log-in form and in a trusted environment.

Note: If you want to use another authentication handler or write your own one, make sure that the handler returns a SimpleCredentials object.

4.4.1.1 Configuring the LDAP Pool

The LDAP authenticator uses the WSM LDAP resource pool. This is switched off by default. If you switch it on, WSM establishes a connection to the LDAP server when it starts.

Creating a Default Initial Context

WSM needs a default initial context to use external resources in the resource pool. If WSM runs on an application server, it uses the application server's default initial context. If it runs outside of an application server, you need to create the default initial context.

The context configuration is stored in the file **resource.xml**, in the folder /bootstrap. To create the default initial context, add an empty <defaultinitialcontext> element as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE resource SYSTEM
"cq:/system/resources/dtd/resource.dtd">

<resource>

  <!-- Define the JNDI initial contexts -->
  <jndicontexts>
    <defaultinitialcontext>
    </defaultinitialcontext>
  </jndicontexts>

  <!-- Define resource pools -->
  <resourcepools>
  ...
  </resourcepools>

</resource>
```

Note: If WSM runs on an application server, you do not need to create a default initial context.

Enabling the LDAP Pool

To enable the LDAP pool, add it to the resources in the file **resource.xml** in the folder /bootstrap, as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE resource SYSTEM
"cq:/system/resources/dtd/resource.dtd">

<resource>
    <jndicontexts>
```

```
        <defaultinitialcontext>
        </defaultinitialcontext>
    </jndicontexts>

    <!-- Define resource pools -->
    <resourcepools>
        <pool name="ldapPool"
config="systemldappool.xml" />
    </resourcepools>
</resource>
```

This creates a new LDAP pool with the name “ldapPool”.

Configuring the LDAP Pool

By default, the LDAP pool is configured in the file **systemldappool.xml** in the folder /bootstrap. The following example configuration connects to an LDAP server using a user name and password:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ldappool SYSTEM
"cq:/system/resources/dtd/ldappool.dtd">

<ldappool>
  <datasource
    name="ldapDirectory"
    host="ldap.myCompany.com"
    port="389"
    authdn="cn=manager,dc=myCompany,dc=com"
    authpw="secret"
  />
</ldappool>
```

name	The name of the data source. If possible, do not change. If you change the name, you have to modify other configuration files that use this name for the LDAP data source.
host	The server that runs the LDAP directory.
port	The port on which the LDAP server answers requests. 389 is the default port for an LDAP server.
authdn	The full “dn” entry of the user account you want to use to read the LDAP user information. Note that you may have to specify the full path of the user, or you cannot connect to the server.
authpw	The password used to log into the LDAP server.

Note: The default configuration file contains a minimal entry for an LDAP pool. Make sure to replace, remove, or comment out that entry. The entry does not work as it is, and may create an error when WSM starts.

Restarting WSM

After you have configured the resource pool, **restart WSM**. The security service, which you will configure shortly, updates its configuration immediately when you change the configuration file. If the resource pool is not available, this will lead to an error.

Note: When you restart, WSM may take more time to load than before. In the authoring and administration environment, you will receive a message that tells you that it is not ready, and you cannot log in with the WSM Development Environment. After a few minutes, the message disappears and you can log in as usual.

4.4.1.2 Enabling the Authenticator

To use the LDAP authenticator, you have to specify it in the file **securityservice.xml**, in the folder /config/repository.

Note: If you have migrated from WSM 3.x, the settings are part of the file contentbus.xml in the /config/contentbus folder.

The authenticator is specified in the <authenticators> element. To use LDAP, uncomment the LDAP authenticator entry, as follows:

```
<authenticators defaultpackage="com.day.cq.contentbus">
    <authenticator
class="com.day.cq.contentbus.driver.simpleldap.LdapAuthen
ticator"
       config="authenticators/ldapauthenticator.xml" />

    <authenticator class="SimpleAuthenticator"
config="simpleauthenticate.xml" />

</authenticators>
```

Put the LDAP authenticator **in front of** the simple authenticator. This way, WSM first tries to find the user name on the LDAP server. If it does not exist, WSM uses the simple authenticator, which authenticates WSM users (such as the “superuser” account). This allows you to log in to both an LDAP user account and a WSM user account.

Note: Make sure that you do not have users on the LDAP server that have the same user name as the WSM system users. For example, if you have a user named “superuser” on the LDAP server, you may not be able to access WSM’s superuser account anymore.

4.4.1.3 Creating WSM LDAP Users and Groups

If you use LDAP, WSM stores the LDAP user and group information as WSM users and groups. In WSM, users and groups are stored as pages, and you must specify the Content Storage Definition (CSD) for the LDAP user page and the LDAP group page.

Creating the LDAP User CSD

To store LDAP user information, WSM requires a user CSD as follows:

- It must be based on the WSM user CSD.
- It must have an atom with the label “dn”, where WSM stores the LDAP dn information.

The LDAP user CSD looks as follows:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE csd SYSTEM
"cq:/system/resources/dtd/xmlcsd.dtd">
<csd name="ldapuser" base="user">
  <hierarchy_driver name="default" />
  <container base="user">
    <atom label="dn" driver="default" />
  </container>
</csd>
```

To create the user CSD, proceed as follows:

1. Open the WSM Development Environment (CQDE).
2. In the **/apps** folder, right-click the folder **Authentication**, and then click **New**. The Create New Page window opens.
3. In the Label field, type **ldapuser.xml**. In the CSD list, click **CSD**. Click OK to create the file.
4. Double-click the newly created file **ldapuser.xml**. In the edit pane, copy the text from above into the file, and then click the **save** icon.

You have now created the LDAP user CSD, where WSM can store LDAP user information.

Note: If you have migrated from a previous WSM version, you may already have an ldapuser CSD in the file ldapuser.xml in the folder /apps/connectors/simpleldap/generic. If the file exists, WSM uses the CSD from it, and you do not need to create a new CSD file.

Extending the LDAP User CSD

The above CSD is a minimal setup for a WSM LDAP user. A user created with this CSD can store the following information:

- WSM stores the user name as the name of the WSM LDAP user page.
- WSM stores the path of the user on the LDAP server (the so-called “dn”) in the atom “dn”.

The LDAP user CSD extends WSM’s default user CSD. In addition to the values specified in the LDAP user CSD, you can use the following values from the default user CSD:

UserID	The user ID.
Password	The password. If you use LDAP authentication, it is not necessary to store the password in WSM.
Fullname	The full name of the user.
Language	The language of the user. WSM uses the authoring environment and spelling checker in this language, if it is available.
EMail	The Email address of the user.
YahooID	The user’s Yahoo! ID.

If you want to have additional information available, you have to extend the CSD so that it can store it. For example, if you store the user’s telephone number on the LDAP server, and you want to use this information in WSM, add the following atom to the LDAP user CSD:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE csd SYSTEM
"cq:/system/resources/dtd/xmlcsd.dtd">
<csd name="ldapuser" base="user">
  <hierarchy_driver name="default" />
  <container base="user">
    <atom label="dn" driver="default" />
    <atom label="phone" driver="default" />
  </container>
</csd>
```

Creating the LDAP Group CSD

To store LDAP group information, WSM requires a group CSD as follows:

- It must be based on the WSM group CSD.

- It must have an atom with the label “dn”, where WSM stores the LDAP dn information.

The LDAP group CSD looks as follows:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE csd SYSTEM
"cq:/system/resources/dtd/xmlcsd.dtd">
<csd name="ldapgroup" base="group">
  <hierarchy_driver name="default" />
  <container base="group">
   <atom label="dn" driver="default" />
  </container>
</csd>
```

To create the group CSD, proceed as follows:

5. Open the WSM Development Environment (CQDE).
6. In the **/apps** folder, right-click the folder **Authentication**, and then click **New**. The Create New Page window opens.
7. In the Label field, type **ldapgroup.xml**. In the CSD list, click **CSD**. Click OK to create the file.
8. Double-click the newly created file **ldapgroup.xml**. In the edit pane, copy the text from above into the file, and then click the **save** icon.

You have now created the LDAP group CSD, where WSM can store LDAP group information.

Note: If you have migrated from a previous WSM version, you may already have an ldapgroup CSD in the file ldapgroup.xml in the folder /apps/connectors/simpleldap/generic. If the file exists, WSM uses the CSD from it, and you do not need to create a new CSD file.

4.4.2 Authentication

The LDAP authenticator uses data from an LDAP server to validate the login information. There are three methods to do so:

- **Read**: The authenticator reads the encrypted password from the LDAP server, encrypts the supplied password, and compares the two values.
- **Compare**: The authenticator encrypts the supplied password and sends it to the LDAP server, which compares it with the stored password.
- **Bind**: The authenticator tries to log into the LDAP server using the user name and password.

We recommend using the **bind** method (which is the default). Read and compare require that you use the same encryption for LDAP and WSM, and are generally more complex to set up, more error-prone and more difficult to maintain.

Finding the User on the LDAP Server

The first step in the LDAP authentication process is that WSM tries to locate an LDAP entry with the user name and password that the user has supplied.

You can configure how WSM communicates with the LDAP server in the file **ldapauthenticator.xml** in the folder /authenticators in the folder /config/repository.

Note: If you have migrated from WSM 3.x, use the file ldapauthenticator.xml in the /config/contentbus folder.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ldapauthenticator SYSTEM
"cq:/system/resources/dtd/authenticators/ldapauthenticato
r.dtd">

<ldapauthenticator>
   <resource poolname="ldapDirectory" />
   <param
      ldapbasedn = "dc=myCompany, dc=com"
      ldapfilter = "objectclass=person"
      ldapaccessmode = "bind"
      ldapuserattributename = "uid" />

   <userfinders>
      ...
   </userfinders>

</ldapauthenticator>
```

ldapbasedn	The base path in the LDAP server where WSM looks for user names. Use this to restrict log-in to a part of the LDAP directory. Note that some LDAP servers can perform user search only from the root.
ldapfilter	You can specify a filter, for example to restrict log-in to persons.
ldapaccessmode	Use **bind**.
ldapuserattributename	The name of the LDAP attribute that stores the user name.

Note: If you use an access mode other than bind, you have to specify additional attributes. Refer to the configuration file DTD for more information.

Authenticating a User

When a user logs in, the LDAP authenticator connects to the LDAP server using the above information, the user name and the password. The following outcomes are possible:

- If the LDAP authenticator can log in, it hands the user name over to WSM, so it can find the corresponding user page (see below).
- If the user does not exist, the authenticator returns nothing. WSM uses the next authenticator to try to validate the user information, or fails if it does not have any more authenticators to use.
- If the user exists, but the LDAP authenticator cannot log in to the LDAP server, it throws a login exception. In this case, WSM does not try other authenticators.

4.4.3 Synchronizing

WSM needs a user page to create a Web page. If you use LDAP, there are two options available:

- You batch-import the LDAP users and groups in fixed time intervals.
- You dynamically create a new user page if a user is legitimate, but does not have a corresponding WSM page.

You have the same options available to synchronize LDAP groups. However, because groups usually have less information associated with them and change less frequently, they are also easier to manage and maintain.

Note: If you create the users dynamically, WSM checks the validity of the user on the LDAP server each time the user logs in. So even if the user pages are not perfectly synchronized, users can not log in to WSM after they have been deleted from the LDAP server.

4.4.4 Batch Synchronizing

In batch synchronization, you can configure WSM to synchronize LDAP users and groups in a set time interval. This is usually more straightforward to set up and more reliable than dynamically synchronizing LDAP users.

Due to the synchronization interval, there is usually a delay from the time when you create the user or group on the LDAP server

until you can use it in WSM. If you batch-synchronize the users manually, you can do so at any time.

4.4.4.1 Setting Up the Synchronization Service

The Synchronization Service runs scheduled LDAP synchronizations. You need to register it in the file **application.xml** in the folder /config. To do so, modify the file as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE application SYSTEM
"cq:/system/resources/dtd/application.dtd">
<application>
  <services defaultpackage="com.day.cq.cms.services">
    <?include /apps/*/config/application/services.xmli
?>
    <service
  class="com.day.cq.cms.services.NotificationService"
      config="cms/notificationservice.xml"
    />
     <service
class="com.day.cq.cms.msm.MultiSiteManagerService"
         config="cms/msm.xml"
     />
  <service
 class="com.day.cq.contentbus.importer.DriverSyncservice"
  config="cms/ldapSync.xml">
  </services>
</application>
```

You can configure the service in the file ldapSync.xml in the folder /config/cms, as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE syncservice SYSTEM
"cq:/system/resources/dtd/drivers/importer/syncservice.dt
d">
<syncservice>
  <driver
class="com.day.cq.contentbus.ldap.importer.UserSynchroniz
ation"
      handle="/etc/ldap/user_import" />
</syncservice>
```

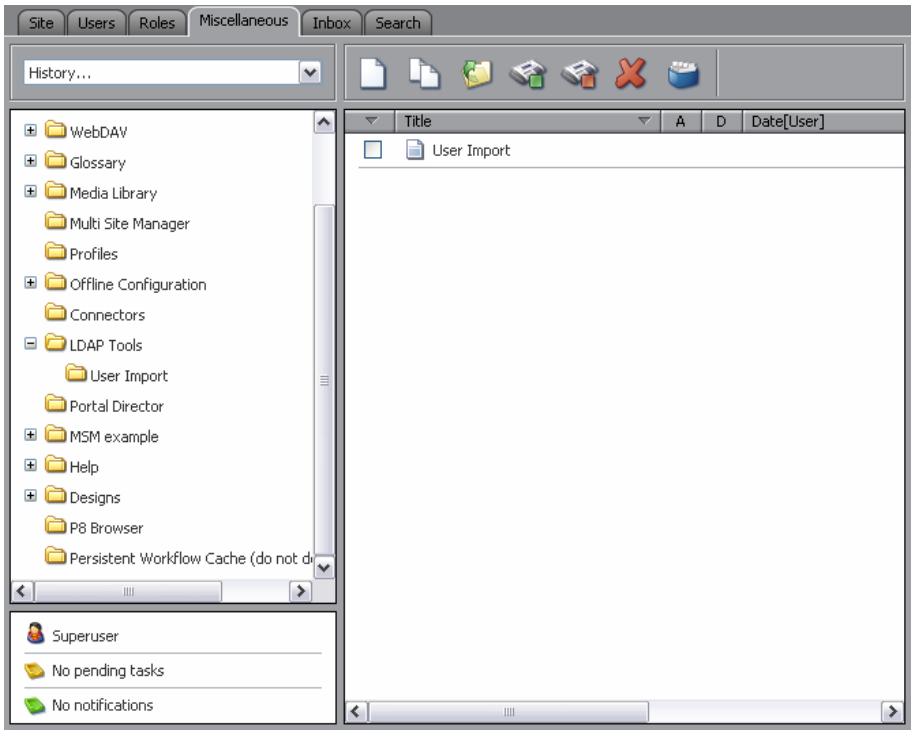
This file points to the page where you will specify the synchronization schedule. The service reads the synchronization configuration from there.

4.4.4.2 The LDAP User Import Configuration Tool

If the LDAP connection is properly set up, you can enable batch synchronization in WSM's authoring and administration environment. To reach the configuration screen, proceed as follows:

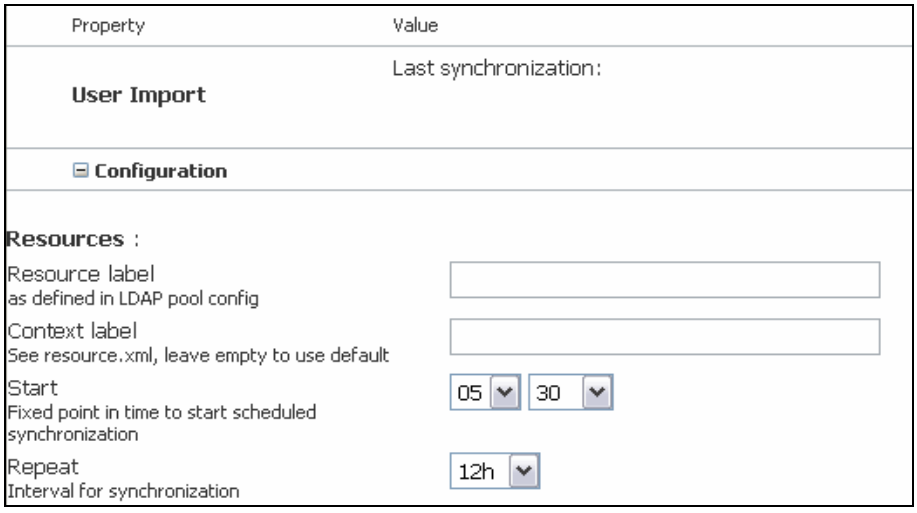
1. Log in to WSM.

2. Click the **Miscellaneous** tab.
3. Click the **LDAP Tools** folder in the left list.
4. Click **User Import**. The LDAP user import tool opens in a new Browser window.



4.4.4.3 Configuring Synchronization

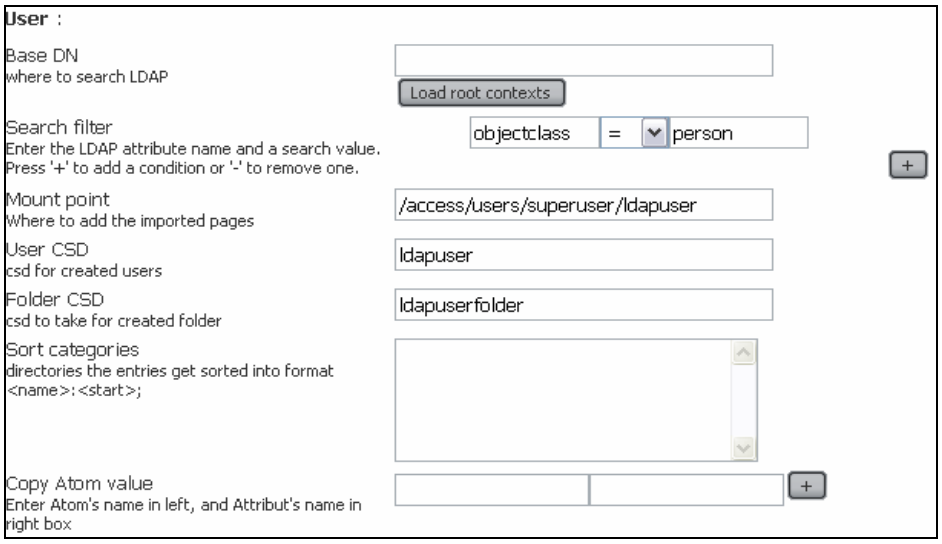
The first paragraph in the LDAP user import tool allows you to set general synchronization parameters (if it is not visible, click the plus sign next to “Configuration”):



Resource label	The name of the LDAP resource pool that WSM uses for LDAP synchronization. If you have configured several resources, WSM lists them, and you can click the one you want to use.
Context label	The label of the initial context used for LDAP synchronization. Leave empty to use the default initial context that is defined in the file resource.xml.
Start	The time of the first synchronization.
Repeat	The interval between synchronizations. The available intervals are fractions of 24 hours, so synchronizing occurs at the same time every day.

For example, if you want to synchronize at 5:30 in the morning and in the evening, set the start time to 05:30 and the repetition interval to 12 hours.

4.4.4.4 Synchronizing Users



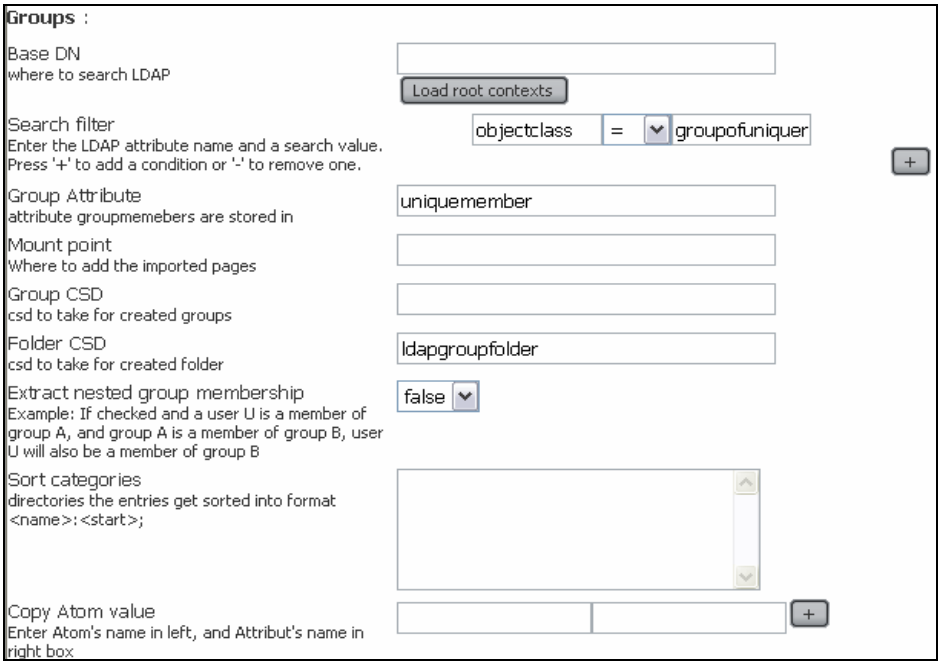
This group allows you to specify how WSM synchronizes users with the LDAP server. You can configure it as follows:

Base dn	The base dn of the LDAP server.
Search filter	A search filter that specifies the LDAP entries WSM searches.
Mount point	The WSM folder that stores the LDAP users. We recommend storing them in the folder **ldapuser** below the superuser folder. If the folder does not exist, WSM creates it.

User CSD	The CSD (Content Storage Definition) WSM uses for LDAP users. You have created this CSD when you have set up the LDAP connection, and by default it is named **ldapuser**.
Folder CSD	The CSD WSM uses for folders that contain LDAP users. To use WSM's user folder for this, type **userfolder**.
Sort categories	If you leave this field empty, WSM re-creates the user hierarchy from the LDAP server. If you specify categories, WSM creates one folder per category, and puts all users that fit into the category into the folder. For example, to create two folders for the users whose name starts with a to m and those whose name starts with n to z, type the following text: a-m:a; n-z:n;
Copy atom value	The LDAP attributes that WSM stores in the WSM LDAP user. By default, type **dn** in both fields, so WSM stores the user's path on the LDAP server on the page. To add more attributes, type the name of the WSM atom in the left field, and the name of the LDAP attribute in the right one. Click the plus sign to add more fields. Note that you have to specify the atoms in the CSD of the WSM LDAP user before you can use them here.

Note: If you want to batch synchronize only groups, you can switch off user synchronization by specifying an invalid search filter. For example, the search filter "objectclass = inexistent" does not return users, and thus switches off batch synchronizing.

4.4.4.5 Synchronizing Groups



Base dn	The base dn of the LDAP server.
Search filter	A search filter that specifies the LDAP entries WSM searches.
Group attribute	The LDAP attribute name that stores group membership.
Mount point	The WSM folder that stores the LDAP users. We recommend storing them in /access/groups/ldapgroups. If the folder does not exist, WSM creates it.
Group CSD	The CSD (Content Storage Definition) WSM uses for LDAP groups. You have created this CSD when you have set up the LDAP connection, and by default it is named **ldapgroup**.
Folder CSD	The CSD WSM uses for folders that contain LDAP users. To use WSM's user folder for this, type **userfolder**.
Sort categories	If you leave this field empty, WSM re-creates the user hierarchy from the LDAP server. If you specify categories, WSM creates one folder per category, and puts all groups that fit into the category into the folder. For example, to create two folders for the groups whose name starts with a to m and those whose name starts with n to z, type the following text: a-m:a; n-z:n;

Copy atom value	The LDAP attributes that WSM stores in the WSM LDAP user. By default, type “dn” in both fields, so WSM stores the user’s path on the LDAP server on the page. To add more attributes, type the name of the WSM atom in the left field, and the name of the LDAP attribute in the right one. Click the plus sign to add more fields. Note that you have to specify the atoms in the CSD of the WSM LDAP user before you can use them here. Typically, add an entry that matches **TitleText** and the LDAP **cn** attribute.

Note: If you want to batch synchronize only users, you can switch off group synchronization by leaving the Mount point field or the CSD field empty.

4.4.4.6 Actions

The following actions are available in the LDAP user import tool:

Save	Saves all changes. Synchronization starts at the time you have specified, and is repeated in the interval you have specified.
Synchronize with LDAP	Loads all users and groups from the LDAP server immediately. WSM displays a list of the synchronized users and groups.
Purge deleted LDAP Users	Removes all WSM LDAP users that exist in WSM, but have been removed from the LDAP server. Note that you cannot log in as these users in WSM as soon as they are removed from the LDAP server.

4.4.4.7 Returning Synchronized Users

The IndexUserFinder returns an existing WSM user page for a user. The IndexUserFinder uses WSM’s internal search engine to find a user page where the “dn” atom matches the requested dn. This is faster and more reliable than using the LdapUserFinder, and does not require that the user hierarchy is the same on the LDAP server and in WSM.

If you do not create LDAP users dynamically, use the IndexUserFinder instead of the LdapUserFinder. Configure this in the file **ldapauthenticator.xml** in the subfolder /authenticators of the folder /repository as follows:

```
...

<userfinders>
    <finder class="IndexUserFinder" />
</userfinders>

...
```

Note: If you have migrated from WSM 3.x, use the file ldapauthenticator.xml in the /config/contentbus folder.

4.4.5 Dynamic Synchronizing

A Communiqué LDAP page stores the “dn” information from LDAP in the “dn” atom. The user resolver uses the dn atom to find user pages, or to create a new page if it does not find one.

WSM uses the LdapDefaultUserFinder to find and create LDAP users in WSM. The LdapDefaultUserFinder is switched on by default. This is configured in the file **ldapauthenticator.xml** in the subfolder /authenticators of the folder /repository.

Note: If you have migrated from WSM 3.x, use the file ldapauthenticator.xml in the subfolder /authenticators of the /contentbus folder.

The default configuration is as follows:

```
  <userfinders>
    <finder class="LdapDefaultUserFinder"
config="ldapdefaultfinder.xml" />
    <finder class="IndexUserFinder" />
  </userfinders>
```

Note: You can remove the IndexUserFinder or leave it. Because the LdapDefaultUserFinder creates the users it does not find, the IndexUserFinder is never used.

4.4.5.1 Synchronizing LDAP Users

The user resolver synchronizes LDAP users and WSM users.

- When a user logs in who exists as a WSM user, the user resolver returns the WSM user.
- If the user exists on the LDAP server, but not in WSM, and you have specified an <autocreate> element, the user resolver creates a new WSM user and returns it.

Note: If you specify <syncatom> elements, WSM re-loads the values of these atoms from the LDAP server every time a user logs in.

Because the user resolver creates pages, you need to configure how to map the LDAP user hierarchy to the WSM user hierarchy. You do this in the file **ldapdefaultfinder.xml** in the subfolder /authenticators of the /contentbus folder. The file looks as follows:

Note: If you have migrated from WSM 3.x, use the file ldapdefaultfinder.xml in the subfolder /authenticators of the /contentbus folder.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE userfinder SYSTEM
"cq:/system/resources/dtd/authenticators/ldapdefaultuserf
inder.dtd">

<userfinder>
   <userresolver>
   <handlemapping type="splitdn"
roothandle="/access/users/superuser/ldap"
basedn="dc=company, dc=com" />

   <autocreate csd="ldapuser">
      <setatom name="dn" attribute="dn" />
      <syncatom name="UserID" attribute="uid" />
      <syncatom name="EMail" attribute="mail" />
      <syncatom name="Fullname" attribute="cn" />
      <syncatom name="Language"
attribute="preferredLanguage" />
      <addgroup handle="/access/groups/author" />
      <addacl type="allow" glob="/content"
rights="rwxcd"/>
   </autocreate>
   </userresolver>
</userfinder>
```

handlemapping	The type attribute specifies how you map users: • **splitdn**: WSM maps the dn to a path, so the WSM user hierarchy reflects the LDAP user hierarchy. • **direct**: WSM creates all LDAP users in one folder. The hierarchy is lost.
roothandle	The path where WSM stores the LDAP users.
basedn	The base path of the LDAP user folder. When WSM maps handles, it removes the basedn information.
autocreate	Creates a new user page automatically. The csd parameter defines the Content Storage Definition (csd) WSM uses for the new page.
setatom	Sets an atom in the new user page. For example, the new page has an atom named “dn” that stores the LDAP dn.

syncatom	Sets an atom in the new user page, and updates the atom content every time a user logs in. This allows you to store user data on the LDAP server and synchronize it with WSM. This function is new in WSM 4.
addgroup	Adds the new user page to a WSM group. This is the default way to assign access rights to a user.
addacl	Adds user rights to the LDAP user. In the example, all LDAP users have full access on the Web site content.

Note: When you delete a user from the LDAP server, it remains in WSM. However, because WSM does not store the user's password, you cannot log in as this user after it is deleted on the LDAP server.

When you synchronize LDAP user attributes, make sure that the atom in which you store the value exists either in the user CSD or the LDAP user CSD. If it does not exist, add it to the LDAP user CSD.

4.4.5.2 Synchronizing Group Relations

The <groupsync> element specifies how WSM finds out to which LDAP groups a user belongs. There are two ways to configure this:

- The <usermemberof> element reads the group membership from the user definition. This is fast, but it works only if the LDAP server stores the group membership in the user information (for example if it uses the objectclass "user" to store the user information).
- The <uniquemembers> element searches the group definitions on the LDAP server to find out which groups contain the user. This takes more time, but works with all servers.

To configure the **<usermemberof>** element, add it to the <groupsync> element as follows:

```
   <groupsync roothandle="/access/groups/ldap"
dnatom="dn">
     <ldapgroups>
       <usermemberof follownested="false" />
     </ldapgroups>
   </groupsync>
```

follownested	Use **true** to include subgroups of the stored groups. This takes more time, and may

	cause a significant performance impact. The default is **false**: WSM uses the groups that are stored directly in the LDAP user definition. If you need to use nested groups, we recommend using batch synchronizing for groups.
attribute	The name of the attribute that stores the membership information on the LDAP server. The default is “memberOf”.
basedn	The folder on the LDAP server where the groups are stored. If you specify this, WSM only uses groups below this folder.

To configure the **<uniquemembers>** element, add it to the <groupsync> element as follows:

```
  <groupsync roothandle="/access/groups/ldap"
dnatom="dn">
    <ldapgroups>
      <uniquemembers basedn="dc=company, dc=com" />
    </ldapgroups>
  </groupsync>
```

basedn	The path where WSM looks for LDAP groups. WSM searches these groups to find out to which the user belongs.
filter	Optional attribute for the search filter on the LDAP server.

Note: When you use the <uniquemembers> element, you cannot use nested LDAP groups. WSM uses only the groups to which the user directly belongs.

4.4.5.3 Synchronizing LDAP Groups

In WSM 4, you can synchronize LDAP groups and WSM groups. The group resolver works like the user resolver:

- If the group relationship synchronization returns that the user belongs to a group, and the group exists in WSM, the group resolver returns the WSM group.
- If the group exists on the LDAP server, but not in WSM, and you have specified an <autocreate> element, the group resolver creates a new WSM group and returns it.

Note: If you specify <syncatom> elements, WSM re-loads the values of these atoms from the LDAP server every time a user who belongs to the group logs in. We recommend not using this feature for groups, as it can have lead to a significant performance impact.

To configure the group resolver, add a <groupresolver> element to the file ldapdefaultfinder.xml in the subfolder /authenticators of the folder /repository, as follows:

```
   <groupresolver>
     <handlemapping type="splitdn"
roothandle="/access/groups/ldap" basedn="dc=company,
dc=com" />
     <autocreate csd="ldapgroup">
       <setatom name="dn" attribute="dn" />
       <setatom name="TitleText" attribute="cn" />
     </autocreate>
   </groupresolver>
```

Configure the group resolver with the same settings as the user resolver. Because the groups typically have less information associated with them as the users, the configuration is typically shorter.

Note: If you use the <syncatom> element for groups, WSM loads the group from the LDAP server each time a user who belongs to the group logs in. This can lead to unnecessary traffic on the LDAP server and a performance impact in WSM.

4.4.6 Mixed Synchronizing

You can combine batch synchronizing and dynamic synchronizing as follows:

- Use batch synchronizing for groups. Groups change rarely, and a user typically belongs to several groups, so dynamically synchronizing groups can be slow.
- Use dynamic synchronizing for users. This means that all updates to the central user database are reflected in WSM immediately.

4.4.7 Managing Access Rights

To manage access rights, you have to specify the rights for each group that WSM loads from the LDAP server. Use the WSM authoring and administration environment for this. Refer to the user guide for details on how to do this.

Note that you have no control over the order of the group memberships of a user. In a standard setup where groups allow actions on a branch of the Web site, this does not matter. However, if you use deny statements to prevent access to a section, the

evaluation order may matter, and you may not be able to control exactly which rights a user has.

4.5 Trusted Authentication

In a trusted environment, WSM does not validate passwords. It trusts that a central security server has already checked the user permissions, and that all requests are valid.

Note: If you enable trusted authentication, you are effectively switching WSM's security mechanisms off. Only do so in a trusted environment, where security is handled centrally, and users cannot access WSM directly.

4.5.1 Configuring the Trusted Authentication Handler

The trusted authentication handler is used in trusted environments, where WSM does not have to check access rights (it can "trust" that all requests are valid, usually because they have been verified by a central server before).

How the Trusted Authentication Handler Works

In a trusted environment, a user can access a protected page as follows:

1. The user logs on to the trusted environment, for example by logging in to Windows.
2. The user requests a protected page.
3. The central security server checks if the user is allowed to see the page. If so, it sends the request to WSM, including the user name (but not necessarily the password).
4. WSM identifies the user. This is used only for personalization and logging. WSM does not check for user authentication or a password.
5. WSM returns the page.

The trustedheaderauth.xml File

The file trustedheaderauth.xml in the folder /delivery contains the setup for the trusted header authenticator. The file specifies how the central server passes on the user information to WSM.

sourcetype	The way in which the central server transmits the user information: • **header**: Using an HTTP header (this is the default). • **cookie**: Using a cookie.

	• **parameter**: Using a request parameter.
sourcename	The name of the HTTP header, cookie or request parameter that stores the user information. The default value is “Authorization”.
format	Use **Basic** if the server transmits the user information according to the “HTTP basic” scheme. In this case, WSM expects the word “Basic” followed by a base64-encoded combination of user name and password (the password is ignored). You can also specify a **regular expression** to extract the user name from the information the central server supplied.
defaultuser	This is the default user that WSM uses if the central server does not supply any user information.

For example, the following file specifies that the central server transmits the user information in the HTTP header “Authorization”, using the “HTTP Basic” authentication scheme. If no use is found, WSM uses the user “anonymous”:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE trustedheaderauth SYSTEM
"cq:/system/resources/dtd/delivery/authenticators/trusted
headerauth.dtd">

<trustedheaderauth
    sourcetype="header"
    sourcename="Authorization"
    format="Basic"
    defaultuser="anonymous"
/>
```

The following file uses the value of the HTTP header “myHeader” as the user name. The regular expression “.*” or “/.*/” means that WSM uses all characters (“.” stands for any character, and “*” stands for any number).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE trustedheaderauth SYSTEM
"cq:/system/resources/dtd/delivery/authenticators/trusted
headerauth.dtd">

<trustedheaderauth
    sourcetype="header"
    sourcename="myHeader"
    format=".*"
    defaultuser="anonymous"
```

```
/>
```

The following file uses any characters up to the character “@” as the user name. For example, if the central server supplies the user information “cmiller@myCompany.com”, WSM extracts “cmiller” as the user name. If the user information does not have an “@” sign, WSM uses the entire user information as the user name.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE trustedheaderauth SYSTEM
"cq:/system/resources/dtd/delivery/authenticators/trusted
headerauth.dtd">

<trustedheaderauth
    sourcetype="header"
    sourcename="myHeader"
    format="/^([^@]+)/"
    defaultuser="anonymous"
/>
```

The following file also matches the email address, but it fails if the address does not end with “@myCompany.com”. In this case, WSM uses the default user. Note that you can use parentheses in the regular expression to use only a part of the match.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE trustedheaderauth SYSTEM
"cq:/system/resources/dtd/delivery/authenticators/trusted
headerauth.dtd">

<trustedheaderauth
    sourcetype="header"
    sourcename="myHeader"
    format="/^([^@]+)@myCompany.com/"
    defaultuser="anonymous"
/>
```

Security

In a trusted setup, WSM does not handle passwords. The security depends on the efficiency of your trusted environment (which is typically very high).

Note: If you use a trusted setup in an environment where users can access WSM directly, security is very low, because users can log in to any user account without specifying a password.

4.5.2 Enabling the Trusted Authentication Handler

Some of WSM’s tools cannot use sessions. To use the trusted authentication handler, you have to define that these tools use the WSM default authentication handler, and that the rest of WSM

(which includes the Web site content) uses the trusted authentication handler.

Excluding Incompatible Areas

Some of WSM's tools require header authentication. If the trusted environment uses a different authentication method, you need to use the default authentication handler for these tools.

In the file **auth.xml**, create one package for each area. Note that you also have to switch off sessions for these areas, because WSM uses them by default if they are enabled. In the package list, remove the current package and specify the following packages (they are already in the file, but commented out):

```
<!-- WSM Development Environment -->
     <package default="none" handler="header" param="WSM
Development Environment" session="false">
         <include glob="/cqde" />
       <include glob="/system/cqde" />
       <include glob="/system/cqjd" />
       <include glob="/libs/TemplateWizard" />
       <include glob="/libs/CFC/content/wizards" />
       <exclude
glob="/libs/CFC/content/wizards/dialog/convertCFCDlg.html
" />
       <include glob="/libs/CFC/content/post.*" />
      </package>

<!-- WebDAV -->
     <package default="none" handler="header" param="WSM
WebDAV">
         <include glob="/etc/webdav" />
      </package>

<!-- Replication -->
     <package default="none" handler="header" param="WSM
Replication" session="false">
       <include glob="/system/replication" />
      </package>

<!-- System installation -->
     <package default="none" handler="header" param="WSM
Installer" session="false">
         <include glob="/system/installer" />
      </package>
```

Note: In this setup, WSM does not check the password that a user types, because it trusts that all requests are valid. Make sure that the central security server checks if users have access to the WSM tools.

Using the Trusted Authentication Handler

Below the packages that use the header authentication handler, add one package that specifies the parameter authentication handler for everything. Because WSM evaluates the packages from top to bottom, this package is used only for the areas that do not match a previous package.

```
    <package default="none" handler="parameter">
        <include glob="*" />
    </package>
```

Switching On the Trusted Authentication Handler

To switch on the trusted authentication handler, you need to add it to WSM's list of authentication handlers. In the folder /delivery, in the file **auth.xml**, add it to the existing handlers as follows:

```
   <handlers defaultpackage="com.day.cq.delivery.auth">

      <handler name="header"
class="AuthorizationHeaderAuthenticator" />

     <handler name="parameter"
class="ParameterAuthenticator"
             config="/config/delivery/parameterauth.xml"
/>

      <handler name="trusted"
class="TrustedHeaderAuthenticator"
config="/config/delivery/trustedheaderauth.xml" />

   </handlers>
```

4.5.3 Enabling the Trusted Authenticator

The trusted authenticator does not check a user's password. It takes the user name that the trusted authentication handler supplies, and returns the corresponding WSM user.

To use the trusted authenticator, add it in the file **securityservice.xml** in the folder **/repository**. Replace the list of authenticators with the trusted authenticator, as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE security SYSTEM
"cq:/system/resources/dtd/securityservice.dtd">
<security>

  ...

  <authenticators defaultpackage="com.day.cq.contentbus">
    <authenticator class="TrustedAuthenticator"/>
  </authenticators>
</security>
```

4.5.4 Enabling Trusted LDAP Authentication

If you use the LDAP authenticator, you need to configure the LDAP authenticator to work with authenticated users. You can do so in the file **ldapauthenticator.xml** in the folder **/config/repository/authenticators**. Add the following attribute to the <param> tag:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ldapauthenticator SYSTEM
"cq:/system/resources/dtd/authenticators/ldapauthenticato
r.dtd">

<ldapauthenticator>
   <resource poolname="ldapDirectory" />

   <param
      ldapbasedn                  = "dc=company, dc=com"
      ldapfilter                  = "objectclass=person"
      ldapaccessmode              = "bind"
      ldapuserattributename       = "uid"
      acceptpreauthenticatedusers = "true" />

  <userfinders>
    ...
  </userfinders>

</ldapauthenticator>
```

Note: If you have migrated from WSM 3.x, use the file ldapauthenticator.xml in the /config/contentbus/authenticators folder.

5 Tools

Some WSM tools rely on configuration files. This section lists the tools that do. Note that other tools are configured in the authoring and administration environment.

5.1 Link Checker

WSM offers two tools to check the link consistency: The external link checker and the internal link checker. Both tools are used differently, and accessed from different places:

- The internal link checker is configured using configuration files. It works automatically for each page WSM serves. If it detects an error, it flags the link in the authoring environment, and removes it in the live environment.
- The external link checker is a tool in the WSM authoring environment. You can configure it there, and start it on a schedule or manually. It compiles a list of the errors it finds.

5.1.1 Internal Link Checker

The link checker locates problems with links, and helps you to manage problematic or broken links. The internal link checker checks the links when WSM serves a page. You do not need to start it manually, or specify a schedule.

Default Settings

By default, the link checker checks all common HTML tags (such as <a>, <area>, <img>) in the following locations:

- /content
- /system
- /libs
- /etc

It removes the invalid links it finds. If it finds a link to a page that is not active yet or not active anymore, it leaves the link, but marks it as

The File linkchecker.xml

This file specifies what happens when WSM discovers an invalid link. It is configured differently for the authoring and the publishing environment, so authors see the invalid links, but Web site visitors do not.

The link checker has the following categories of error:

- Predated: The page exists, but is not published yet.
- Expired: The page exists, but is not published anymore.
- Invalid: The page does not exist.

On the publishing environment, the file linkchecker.xml looks as follows. It specifies that all invalid links are removed.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE linkchecker SYSTEM
"cq:/system/resources/dtd/delivery/rewriter/linkchecker.d
td">

<linkchecker>

   <expired remove="true">
   </expired>

   <predated remove="true">
   </predated>

   <invalid remove="true">
   </invalid>

</linkchecker>
```

On the authoring environment, the file looks as follows. It specifies that authors can click links to pages that are outdated or not live yet, while invalid links are removed. Also, WSM displays an icon before and after each faulty link.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE linkchecker SYSTEM
"cq:/system/resources/dtd/delivery/rewriter/linkchecker.d
td">

<linkchecker>
  <expired remove="false">
    <prefix>
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_o.gif
alt='expired link: %s' title='expired link: %s'
border=0>]]>
    </prefix>
    <suffix>
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_c.gif
border=0>]]>
    </suffix>
  </expired>
  <predated remove="false">
    <prefix>
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_o.gif
alt='predated link: %s' title='predated link: %s'
border=0>]]>
    </prefix>
    <suffix>
```

```
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_c.gif
border=0>]]>
    </suffix>
  </predated>
  <invalid remove="true">
    <prefix>
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_o.gif
alt='invalid link: %s' title='invalid link: %s'
border=0>]]>
    </prefix>
    <suffix>
      <![CDATA[<img
src=/libs/CFC/content/statics.linkcheck_c.gif
border=0>]]>
    </suffix>
  </invalid>
</linkchecker>
```

The File mapper.xml

This file maps ContentBus handles to URLs and back. Internally, WSM uses a ContentBus handle to point to a page, while URLs work on the Web. WSM uses the server URL to map the paths automatically, the file mapper.xml specifies which parts of the ContentBus are mapped, and in which direction.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mapper SYSTEM
"cq:/system/resources/dtd/delivery/mapper.dtd">

<mapper>
  <fakeurls>
  ...
  </fakeurls>

  <mapping direct="true">
    <map from="/content/" to="/" />
   <map from="/apps/*/docroot/" to="/" />
    <map from="/libs/*/docroot/" to="/" />
    <map from="/system/docroot/" to="/" />
    <!-- the following is a workaround for a bug in the
Windows XP WebDAV redirector -->
    <map from="/etc/" to="/etc//" direction="inwards" />
  </mapping>
</mapper>
```

In the file mapper.xml, the <mapping> section specifies the mapping. If you set the attribute **direct** to **true**, WSM accepts requests that directly supply a ContentBus handle. If you set it to **false**, WSM maps all requests.

Each <map> element specifies a mapping from a ContentBus handle to a URL. By default, you do not have to change the mapping for link checking.

The File rewriter.xml

When WSM renders a page, the rewriter replaces all ContentBus handles with URLs. This file specifies which html tags and attributes the rewriter checks.

The rewriter is also used to check the internal links. If you use advanced or non-standard html code, make sure that the tags list in the file rewriter.xml covers all the tags you use for links. If you use a different tag, add it to the tag list.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rewriter SYSTEM
"cq:/system/resources/dtd/delivery/rewriter.dtd">

<rewriter globalCallbacksOnInclude="true">
  <tag name="a" link="href" endtag="true" />
  <tag name="area" link="href" />
  <tag name="table" link="background" />
  <tag name="tr" link="background" />
  <tag name="td" link="background" />
  <tag name="img" link="src" />
  <tag name="link" link="href" />
  <tag name="frame" link="src" />
  <tag name="iframe" link="src" />
  <tag name="input" link="src" />
  <tag name="form" link="action" />
  <tag name="script" link="src" endtag="true" />
  <callback config="rewriter/linkchecker.xml"
class="com.day.cq.delivery.linkchecker.DefaultLinkChecker
">
    <package>
      <include glob="/content/*" />
      <include glob="/system/*" />
      <include glob="/libs/*" />
      <include glob="/etc/*" />
    </package>
  </callback>
</rewriter>
```

5.1.2 External Link Checker

Refer to the WSM User Guide for information about how to set up and use the external link checker.

5.2 WSM JMX Console

The file management.xml contains the setup for the WSM JMX management console, which allows you access to the Java Bean administration of an authoring instance. By default, the management console is available at port 8090, with user name and password set to “superuser”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE management SYSTEM
"cq:/system/resources/dtd/management.dtd">

<management domain="Communique4"
mBeanServerType="private">

    <http port="8090">
        <user name="superuser" password="superuser" />
    </http>

</management>
```

Note: For security reasons, change the password and/or use a firewall to block access to port 8090 in a productive environment.

6 Performance

The following settings allow you to tweak WSM's performance settings. The following factors typically have a large impact on performance:

- Increasing the memory available for the virtual machine (see below).
- Using the dispatcher to cache WSM pages. See the dispatcher documentation for details.
- Designing the Web site to take maximum advantage of WSM's internal cache and of the dispatcher. See the dispatcher documentation for details.
- Designing the Web site to take maximum advantage of WSM's internal search engine.

Note: Improper use of the internal search engine can lead to a much worse performance than if you did not use the search engine at all.

6.1 Memory

Using the WSM manager, you can change the amount of memory the Java virtual machine uses for WSM. The default start configuration for WSM 4 is as follows:

```
-Xrs -Xms128m -Xmx256m -jar bin/bootstrap.jar
```

This sets the minimum memory available to 128 megabytes, the maximum to 256 megabytes. For example, to double the memory available, proceed as follows:

1. Open the WSM manager.
2. Double-click the WSM server whose memory settings you want to change.
3. In the VM options field, type the new virtual machine options. For double memory, replace the memory settings with –Xms256m –Xmx512m, respectively.

Note: If you start the virtual machine with a command instead of the WSM manager, you can use the same settings as command-line parameters.

6.2 Caching

WSM uses caching to improve the performance of a Web site. If two users request the same page shortly after one another, WSM does not re-compose the page from scratch, but uses the previously served page again for the second user.

Note: The cache settings have only a limited effect on cache performance. The biggest factor is typically whether your Web site is designed to take advantage of WSM's caching mechanisms.

Default Settings

By default, WSM caches everything except Java classes.

The cache.xml File

The cache.xml file defines the Web site structure for caching. Most settings are used for WSM's internal functions.

```
<cache
   root="/system/work/cache"
   burstmode="true"
   defaultexpiry="-1"
   respectcachecontrol="false"
   denySessions="false"
   csd="cache">

   <deny>
      <cookie name="Show" />
   </deny>

   <!-- allow>
      <cookie name="cq3session" />
   </allow -->

   <packages>
      <package default="none">
         <include glob="/*" />
         <exclude glob="/system/work/classes/*" />
         <exclude glob="/classes/*" />
         <exclude glob="/system/cqjd/*" />
         <types>
              <type glob="text/*" />
              <type glob="image/*" />
              <type glob="application/x-javascript" />
         </types>
         <ignoredeps>
              <include glob="/libs" />
              <include glob="/access" />
              <!-- ignore JSP java sources and classes -->
              <include glob="/classes/*$jsp.*" />
              <!-- ignore ECMA class files -->
              <include glob="/system/work/classes" />
         </ignoredeps>
      </package>
      </packages>

</cache>
```

Setting Burst Mode

If the **burstmode** attribute is “false”, WSM checks the access rights and link integrity for pages in the cache. If “false”, WSM returns pages from the cache more quickly, but pages may have inaccurate links, and users may in some cases access pages they are not allowed to see.

By default, burst mode is enabled. If you have highly sensitive information on the Web site, frequently change the site structure, or if users complain about broken links, set burst mode to false to have WSM check the pages before serving them.

Setting the Cache Expiry

The **defaultexpiry** attribute sets the cache expiry in milliseconds. By default, the attribute is set to “-1”, so WSM does not expire items from the cache after a set time. This means that items are removed only based on WSM’s cache update rules.

Set a cache expiry if you use items that bypass WSM’s cache rules.

Allowing Browsers to Refresh the Cache

Browsers may request a fresh (uncached) page from WSM, for example if a user reloads a page by pressing CTRL and clicking the reload button. By default, WSM ignores such requests and uses the page from the cache.

If you want to allow users to override WSM’s cache rules and request uncached pages, set the **respectcachecontrol** attribute to **true**.

Bad Caching

In some scenarios, caching can negatively affect performance. Two such scenarios are:

- If the Web site is updated frequently, such as several times a minute. In such cases, the cache administration overhead may outweigh the small gain of serving the page repeatedly between updates.
- If the Web site is heavily cross-linked, for example if you integrate a dynamic navigation from every page to every other page in the page code. In this case, every update deletes the entire cache. For this example, we recommend that you use a separate JavaScript file that stores the navigation content.

6.3 Search

WSM uses the search function for both Web site searches and for its internal search tasks. Optimizing search can dramatically improve WSM's overall performance.

Default Settings

By default, WSM uses the Lucene search engine for the ContentBus /content folder and its internal search engine for everything else.

The searchservice.xml File

The searchservice.xml file specifies the search services WSM uses. For each service, it specifies the search class, the configuration file and which parts of the ContentBus it searches.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE searchservice SYSTEM
"cq:/system/resources/dtd/contentbus/searchservice.dtd">
<searchservice>

   <handler class="com.day.cq.search.standard.Search"
config="search/defaulthandler.xml">
     <index>
       <package default="none">
            <include glob="*" />
            <exclude glob="/content/*" />
            <exclude glob="/system/work/classes/*" />
            <exclude glob="/classes/*" />
            <exclude glob="/system/bin/*" />
            <exclude glob="/system/bin.*/*" />
       </package>
     </index>
   </handler>

   <handler
class="com.day.cq.search.fulltext.LuceneSearch"
config="search/fulltexthandler.xml">
     <index>
       <package default="none">
            <include glob="/content/*" />
       </package>
     </index>
   </handler>

</searchservice>
```

The defaulthandler.xml File

This is the configuration file for WSM's internal search engine. The configuration contains the name of the search database file and the block size.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE search SYSTEM
"cq:/system/resources/dtd/contentbus/search.dtd">
<search>
  <database
    name="search.db"
    blocksize="128"
  />
</search>
```

The block size defines how big one block of information is. Large blocks mean that WSM uses more memory on blocks that are filled only partially. Small blocks mean that WSM has to use multiple blocks to store one piece of information. If the block size is too small, you may receive an “out of inodes” error when WSM cannot add more blocks.

The fulltexthandler.xml File

This is the configuration file for the Lucene search engine that searches the Web site content.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE searchhandler SYSTEM
"cq:/system/resources/dtd/contentbus/search/fulltexthandl
er.dtd">
<searchhandler>
  <index location="search/fulltext"
         maxFieldLength="10000"
         resultCacheSize="50"
         useCompoundFile="true" />
</searchhandler>
```

The attribute **maxFieldLength** sets the maximum number of terms Lucene indexes per document. For a Web site, this value is typically sufficient. If you have large text files with more than 20’000 words, you may have to increase the value. To check, see if your search finds words that occur only at the end of your larger documents. Note that increasing the value increases memory consumption.

The attribute **resultCacheSize** specifies how many result sets Lucene keeps in its internal cache. If Lucene caches the result sets, you can pose the same query repeatedly without a performance impact. This means that you do not have to write your own code to cache a user’s search results.

7 System Architecture

This section tells you how to set up WSM to provide a safe and reliable infrastructure.

7.1 High Availability

WSM stores the repository data in a CRX repository. For information on setting up a high-availability CRX solution, refer to the CRX Configuration Guide. After you have installed a high-availability CRX solution, you can install WSM on top of it, so your WSM solution is also protected against hardware and software failure.

7.2 Hot Backup

WSM stores its content in a CRX repository, which in turn uses an internal or external database for persistent storage. If you use a database that supports hot backup, you can use the database's backup functions to create a backup while WSM runs. Refer to the database documentation for further information.

8 WSM Servlet Engine

A servlet engine is a program that runs Java servlets. WSM can run in its own servlet engine, or you can run it in a third-party servlet engine. If you use the WSM Servlet Engine, the administration tool allows you to:

- Add and remove (“deploy” and “undeploy”) Web applications.
- Start and stop Web applications.
- Change the administrator password of the servlet engine.
- See the Java Virtual Machine information.

8.1 Connecting

When you install the WSM Servlet Engine, you can specify its context and the administrator user name and password. The context is the location at which you can reach the administrator screen. If you keep the default context of /admin, you can reach the WSM Servlet Engine by typing **www.myServer.com/admin/** in the address field of your browser.

You need to log in using the WSM Servlet Engine administrator user name and password. If you have kept the default user name and password during installation, we strongly recommend that you change the administrator password now. The default user name is “admin” and the default password is “admin” as well.

8.2 Adding a Web Application

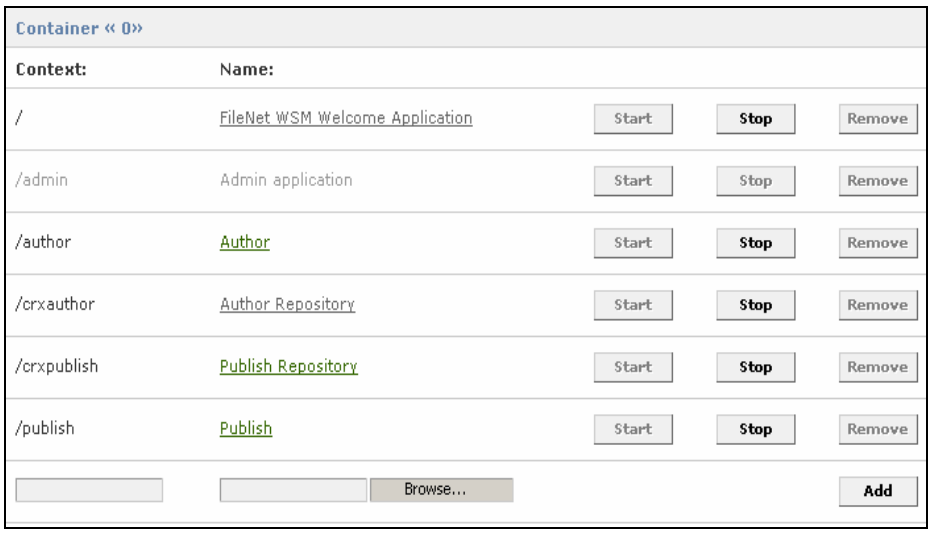
To add a new Web application to the Servlet Engine, proceed as follows:

1. In the **Context** field, type the context for the new Web application. For example, type "/myWebApplication" to make the Web application available in the /myWebApplication folder on the server.
2. Click **Browse** to select the Web application file. This is a file with the extension ".war" that contains the new Web application.
3. Click **Add** to add the new Web application to the server.

The administrator now adds the Web application to the server. This process is also known as "deploying". The Web application file is copied to the server and the server configuration files are modified. You can now reach the Web application using the server address and the Web application context, for example:

www.myServer.com/myWebApplication

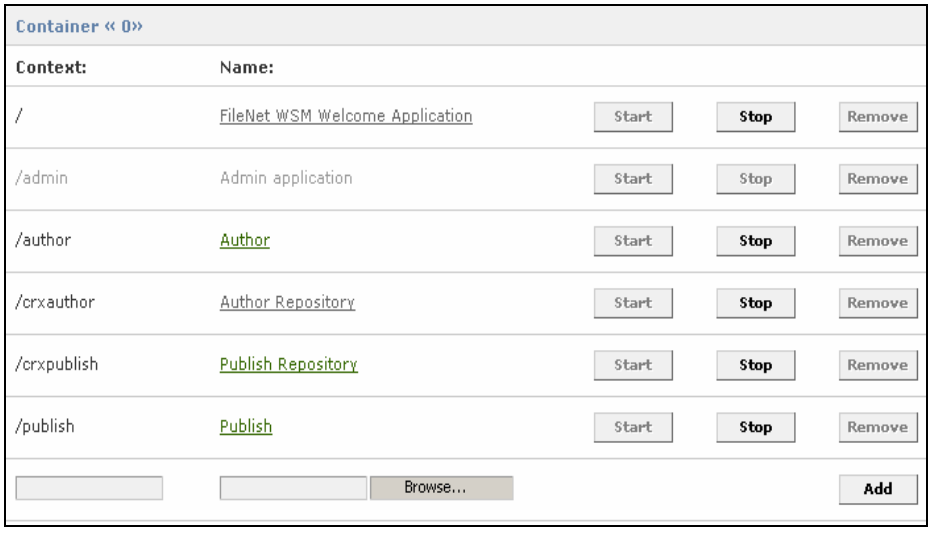
8.3 Removing a Web Application



To remove a Web application from the Servlet Engine, proceed as follows:

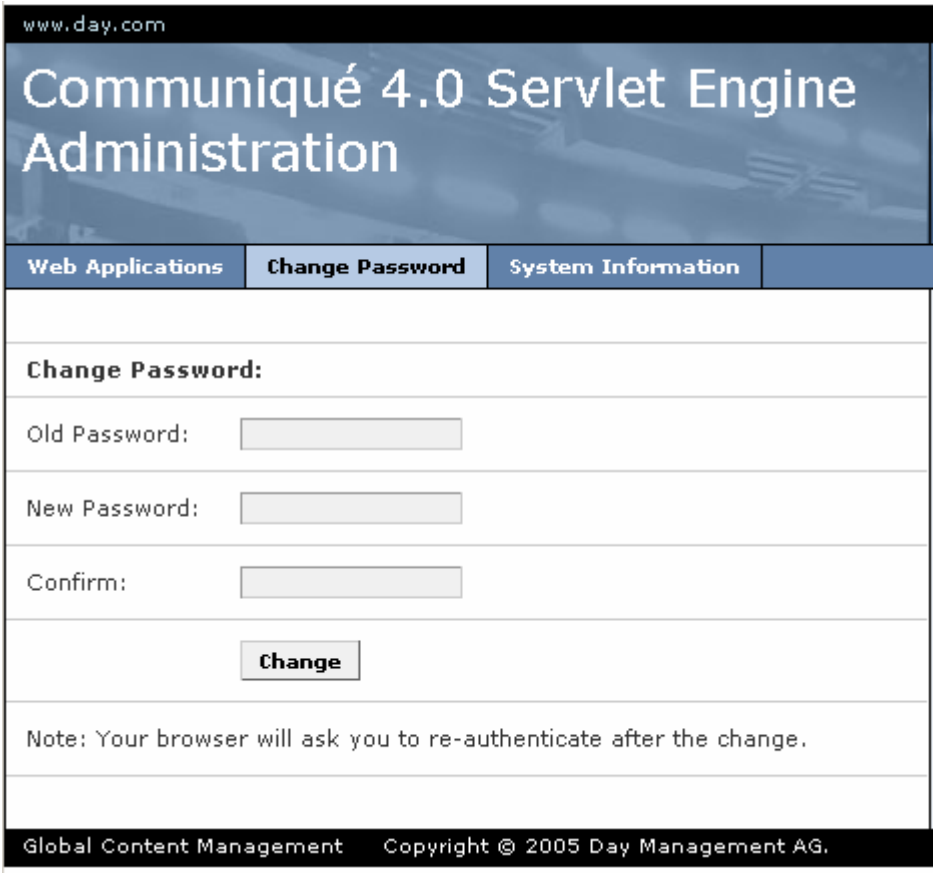
1. Stop the Web application you want to remove by clicking the **Stop** button in the row of the Web application.
2. Click the **Remove** button next to the Web application to remove it.

8.4 Starting and Stopping Web Applications



This page lists the currently installed Web applications. Click **Start** to start a Web application, or **Stop** to stop one. Note that the Web applications reload their configuration files when you stop and restart them. If you have changed the Web application or its configuration, this screen may provide a faster way to make the changes effective than restarting the entire WSM application.

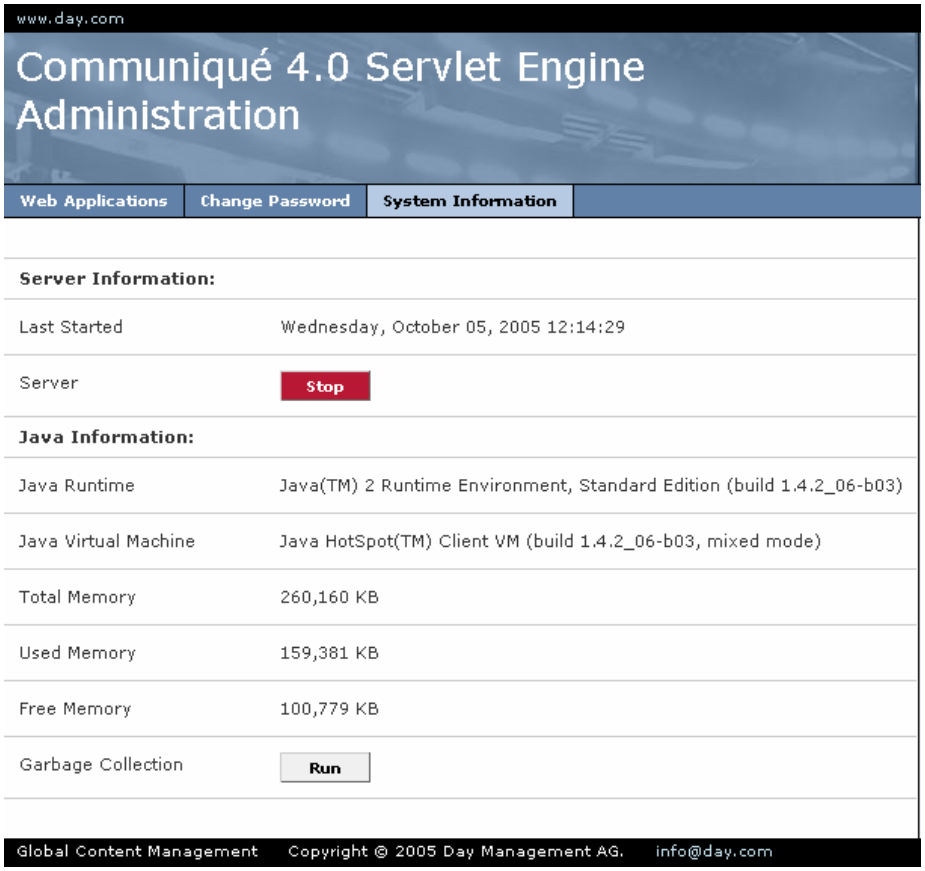
8.5 Changing the Administrator Password



This tab allows you to change the Servlet Engine Administrator password:

1. Click the **Change Password** tab.
2. Type the old password and the new password (twice). Click **Change** to change the password.
3. Log in again with the new password.

8.6 Java Virtual Machine Information



This tab lists the following information:

- The Java version
- The Java Virtual Machine version
- The amount of memory that is used and that is still free and the total amount of memory.

Click **Run** to run the garbage collector of the Java Virtual Machine. The garbage collector will try to free as much unused memory as possible.

Click **Stop** to stop the server.