Adobe

Administering LiveCycle® Content Services 9

Adobe® LiveCycle ES2

September 24, 2010

Version 9.0

Contents

About This Document

This document provides administrators with details about the daily, weekly, and monthly administrative tasks required to keep the Adobe® LiveCycle® ES2 (Enterprise Suite 2) installation running smoothly.

This document does not include installation, configuration, or deployment information; this information is documented in the *Installing and Deploying LiveCycle ES2* document for your application server or in *Installing and Deploying LiveCycle ES2 for JBoss Using Turnkey*.

This document does not include extensive information about using the LiveCycle Administration Console to configure system settings; this information is documented in the *LiveCycle Administration Console Help,* available within LiveCycle Administration Console".

Who should read this document?

This document provides information for IT and product administrators:

IT administrator: Responsible for IT deployment planning and hardware preparation. Knowledgeable about application servers, LDAP, database and network administration.

Product administrator: Responsible for installing, monitoring, maintaining, and troubleshooting a multiservice LiveCycle ES2 environment. This LiveCycle ES2 administrator will work with the IT administrator before installing the LiveCycle ES2 software into the corporate network.

The information provided is based on the assumption that anyone reading this guide is familiar with J2EE application servers, Linux®, and Microsoft® Windows®, IBM® AIX®, or Sun™ Solaris™ operating systems, MySQL, Oracle®, DB2®, or SQL Server database servers, and web environments.

Conventions used in this document

This document uses the following naming conventions for common file paths.

Name	**Default value**	**Description**
[LiveCycleES2 root]	Windows: C:\Adobe\Adobe LiveCycle ES2\ Linux and UNIX: /opt/adobe/adobe_livecycle_es2/	The installation directory that is used for all LiveCycle ES2 solution components. This directory contains subdirectories for LiveCycle Configuration Manager, the LiveCycle ES2 SDK, and each LiveCycle ES2 solution component that is installed (along with the product documentation). This directory also includes directories that relate to third-party technologies.
[appserver root]	JBoss 4.2.0 or 4.2.1 on Windows: C:\jboss JBoss 4.2.0 or 4.2.1 on Linux, Solaris: /opt/jboss JBoss Enterprise Application Platform 4.3 on Windows: C:\jboss-eap-4.3\jboss-as JBoss Enterprise Application Platform 4.3 on Linux, Solaris: /opt/jboss-eap-4.3/jboss-as WebSphere on Windows: C:\Program Files\IBM\WebSphere\AppServer WebSphere on Linux and Solaris: /opt/IBM/WebSphere/AppServer WebSphere on AIX: /usr/IBM/WebSphere/AppServer, or, /opt/IBM/WebSphere/AppServer WebLogic on Windows: C:\bea\wlserver_10.3 WebLogic on Linux and UNIX: /opt/bea/wlserver_10.3	The home directory of the application server that runs the LiveCycle ES2 services.
[appserverdomain]	WebLogic on Windows: C:\bea\user_projects\domains\base_domain WebLogic on Linux and UNIX: /opt/bea/user_projects/domains/base_domain	The domain you configured on WebLogic.
[dbserver root]	Depends on the database type and your specification during installation.	The location where the LiveCycle ES2 database server is installed.

Most of the information about directory locations in this guide is cross-platform (all file names and paths are case-sensitive on Linux and UNIX®). Any platform-specific information is indicated as required.

All references to beans in this guide are to Spring beans.

Additional information

The resources in this table can help you learn aboutLiveCycle ES2.

For information about	See
General information about LiveCycle ES2 and the solution components	*LiveCycle ES2 Overview*
What's new in the Adobe LiveCycle ES2 (Enterprise Suite) release	*What's New for LiveCycle ES2*
LiveCycle ES2 terminology	*LiveCycle ES2 Glossary*
Other Adobe LiveCycle ES2 solution components	*www.adobe.com/products/livecycle*
Other services and products that integrate with LiveCycle ES2	*partners.adobe.com/public/developer/main.html*
Installing Adobe LiveCycle Workbench ES2	*Installing Your Development Environment*
Upgrading to LiveCycle ES2 from a previous version.	*Preparing for Upgrading to LiveCycle ES2* *Upgrading to LiveCycle ES2 for JBoss* *Upgrading to LiveCycle ES2 for WebSphere* *Upgrading to LiveCycle ES2 for WebLogic*
All the documentation available for LiveCycle ES2	*Adobe LiveCycle ES2 Documentation*
LiveCycle ES2 release information and last-minute changes that occur to the product	*Release Notes*
Patch updates, technical notes, and additional information about this product version	*LiveCycle Technical Support*

1 Customizing Content Services 9

This chapter describes the tasks that are required to customize Adobe LiveCycle Content Services 9, the solution component that brings content management capabilities to your LiveCycle ES2 environment.

This document is organized into the following sections:

1.1 Installing Content Services 9 components

1.1.1 Installing Flash Player

This step is optional and may be completed later.

1. Browse to the Adobe Flash Player download website: http://www.adobe.com/products/flashplayer
2. Download the latest (stable) version of Flash Player for your platform.
3. Browse to the location of your downloaded file and install the application.

   A wizard guides you through the installation.
4. When the installation is complete, click Close.

1.1.2 Installing an AMP

Refer to the "Configuring and deploying LiveCycle ES2" section in the *Installing and Deploying LiveCycle ES2* guide for your application server.

1.2 Configuring the email service

1.2.1 Configuring the inbound email service

Add the following JVM arguments to the application server startup script to configure the inbound email service:

```
-Demail.server.enabled=true
-Demail.inbound.enabled=true
```

Note: You can also set any other required JVM arguments in the application server startup script.

Alternatively, you can configure the inbound email service by modifying the relevant `.properties` file.

1. Open the contentservices.war\WEB-INF\classes\alfresco\emailserver\email-server.properties file.
2. Modify the behavior of the inbound email service in this file.

   The following table provides configuration examples:

Value	**Description**
`email.inbound.enabled=true`	Enables or disables the inbound email service. The service could be used by processes other than the email server (for example, direct RMI access), so this flag is independent of the email service.
`email.inbound.unknownUser=anonymous`	Specifies the user name to authenticate as when the sender address is not recognized.
`email.server.enabled=true` `email.server.port=25` `email.server.domain=alfresco.com`	Specifies email server properties.
`email.server.allowed.senders=`	Provides a comma-separated list of email REGEX patterns of allowed senders. If there are any values in the list, then all sender email addresses must match. For example: `.*\@alfresco\.com, .*\@alfresco\.org`
`email.server.blocked.senders=`	Provides a comma-separated list of email REGEX patterns of blocked senders. If the sender email address matches this, then the message will be rejected. For example: `.*\@hotmail\.com, .*\@googlemail\.com`

1.2.2 Configuring the outbound email service

The outbound email service is used to send notifications for email-based reviews. Follow these steps to configure the outbound email service:

1. In LiveCycle Administration Console, click **Home > Services > Applications and Services > Service Management.**
2. Click **Email Service** from the list.

3. In the **Configuration** tab, specify details about the mail server. Specify these essential settings:

   SMTP Host: The host name of the mail server to which results and error messages are sent. For example, `namail.<yourorganization>.com`.

   SMTP Port Number: The default value for the SMTP port is 25.

   SMTP User: The user account using which email notifications of results and errors are sent. For example, `tbuser`.

   SMTP Password: The password for the SMTP account. Some mail servers do not require an SMTP password.

   STMP Transport Security: Enable or disable this setting.

   POP3/IMAP: Choose a protocol for the mail server.

1.3 Configuring FTP access

Add the following JVM arguments to the application server startup script to configure FTP access:

```
-Dftp.enabled=true
-Dftp.port=<port number>
```

Alternatively, you can configure FTP access by modifying the following `.properties` file:

```
contentservices.war\WEB-INF\classes\alfresco\file-servers.properties
```

1.4 Disabling the propagation of Content Services 9 events to LiveCycle ES2

Add the following JVM argument to the application server startup script:

```
-DpropagateEventsToLC=false
```

Alternatively, modify the following `.properties` file:

```
contentservices.war\WEB-INF\classes\alfresco\extension\
custom-repository.properties
```

1.5 Setting the usage quota

Add the following JVM arguments to the application server startup script:

```
-Dsystem.usages.enableQuotaSize=true
-Dsystem.usages.quota=<size in KB>
```

Alternatively, modify the following `.properties` file:

```
contentservices.war\WEB-INF\classes\alfresco\extension\
custom-repository.properties
```

1.6 Configuring the audit setting

- To enable auditing, add the following JVM argument to the application server startup script:

  ```
  -Dcontentservices.audit.config=alfresco/extension/auditConfigON.xml
  ```

- To disable auditing, add the following JVM argument to the application server startup script:

  ```
  -Dcontentservices.audit.config=alfresco/extension/auditConfigOFF.xml
  ```

Alternatively, you can enable/disable auditing by modifying the following `.properties` file:

```
contentservices.war\WEB-INF\classes\alfresco\extension\
custom-repository.properties
```

1.7 Configuring the minimum number of search characters

Follow these steps to configure the minimum number of characters that a user can search for:

1. Create the `web-client-config-custom.xml` file in the following format:

   ```
   <alfresco-config>
    <config>
    <client>
    <search-minimum>1</search-minimum>
    </client>
    </config>
   </alfresco-config>
   ```

2. Replace `2` in `<search-minimum>2</search-minimum>` to the minimum number of search characters that you want to set. For example, set `1` for Japanese and `3` for other languages.
3. Login to Contentspace 9 and add this new XML file to the `Company Home > Data Dictionary > Web Client Extension` folder.
4. Open the following URL in a separate browser window:

   ```
   http://<machineIP>:<Port>/contentspace/faces/jsp/admin/
   webclientconfig-console.jsp
   ```

5. Type **reload** in the **Command** box and then click **Submit.**

If you need to change the value for the `<search-minimum>` parameter later, repeat steps 2 to 5.

1.8 Disabling content indexing

To improve Content Services 9 performance, you can disable content indexing. However, disabling indexing will also disable text-based search within new content. Follow these steps to disable indexing:

1. In the adobe-contentservices.ear file, navigate to LiveCycle Content Services.ear/contentservices.war/WEB-INF/classes/alfresco/model and open the contentModel.xml file for editing.
2. Locate the following line:

   ```
   <type name="cm:content">
   ```

3. Set the `index enabled` and `tokenized` properties to `false`.
   - Change `<index enabled="true">` to `<index enabled="false">.`
   - Change `<tokenized>true </tokenised>` to `<tokenized>false </tokenised>.`

1.8.1 Disabling the conversions required for indexing

To realize additional performance improvements, disable the conversions required for indexing. Follow these steps:

1. In adobe-contentservices.ear, browse to contentservices.war\WEB-INF\classes\alfresco\extension.
2. Preserve a backup of the custom-metadata-extractors-context file.
3. Delete this file from the EAR.

2 Administering Content Services 9

This chapter describes the following tasks required to administer Content Services 9:

2.1 Upgrading Content Services 9

Refer to the *Upgrading to LiveCycle ES2* documentation.

2.2 Backing up and restoring

Refer to *Back up and recovery strategy for LiveCycle ES2* in LiveCycle ES2 Administration Help (HTML).

2.3 Monitoring Content Services 9

This section describes how to monitor Content Services 9 using the Java Management Extension (JMX).

2.3.1 Configuring a JMX interface

By default, you can reconfigure Content Services 9 by shutting down the server, editing the relevant property and configuration files and then restarting the server. However, there are some support operations that should be performed on-demand at runtime without needing to restart the server. For example, temporarily changing log levels in order to debug or troubleshoot a live system.

The Java Management Extension (JMX) interface allows you to access Content Services 9 through a standard JMX console that supports JMX Remoting (JSR-160). This lets you:

- Change log levels
- Enable or disable file servers (FTP/CIFS/NFS)
- Set server read-only mode
- Set server single-user mode
- Set server maximum user limit - including ability to prevent further logins
- Count user sessions/tickets
- User session/ticket invalidation

Example consoles include:

- JConsole (supplied with Java SE 5.0 and higher)

- MC4J
- JManage

Some of these consoles also provide basic graphs and/or alerts for monitoring JMX-managed attributes.

2.3.2 Connecting to Content Services 9 through a JMX console/JSR-160

You can connect to Content Services 9 through a JMX console that supports JSR-160.

1. Open a JMX console that supports JMX Remoting (JSR-160).
2. Enter the JMX URL:

   ```
   service:jmx:rmi:///jndi/rmi://<hostname>:50500/alfresco/jmxrmi
   ```

   Where `<hostname>` is the name of your host or IP address.
3. Enter the default JMX user name: `controlRole`
4. Enter the default JMX password: `change_asap`

Note: You must change the default JMX password as soon as possible.

5. Change the JMX password in the following files:
   - `<configRoot>/alfresco/alfresco-jmxrmi.access`
   - `<configRoot>/alfresco/alfresco-jmxrmi.password`

2.3.3 Configuring JMX properties

When you make changes through the JMX interface, such as changing a log level, these changes are lost after the server restarts. To set permanent changes, you can apply the change to the appropriate configuration file.

- To configure the file server:

Attribute	Value
`ftpEnabled`	`true = enable FTP server, false = disable FTP server`
`cifsEnabled`	`true = enable CIFS server, false = disable CIFS server`
`nfsEnabled`	`true = enable NFS server, false = disable NFS server`

This is not cluster-aware. If more than one file server is running (for example, load-balanced FTP), you must apply the change to each machine. Some consoles (for example, JManage) may provide basic facilities for accessing each machine in an application cluster.

- To configure the repository server:

Attribute	Value
`readOnly`	Set repository transaction mode. true =READONLY, false = WRITABLE
`singleUserOnly`	Set single user name, for example, “admin” or blank to disable single user mode and allow all user names.
`maxUsers`	Limit for non-expired user logins. -1 if no limit set, 0 to prevent further logins.
`linkValidationDisabled`	Disable or enable link validation service.

These managed attributes/operations are cluster-aware.

2.3.4 JMX monitoring and management extensions

This section describes the JMX-based monitoring and management functionality.

The monitoring and management extensions can be subdivided into three categories:

Read-only monitoring beans: Expose a variety of real-time metrics for monitoring health and throughput of your Content Services 9 server.

Configuration beans: Provide an easily navigable view of key system configuration for support and diagnostic purposes.

Management beans: Allow control over various subsystems.

Coexistence with other MBeans

If there is an MBean server already running on the Java Virtual Machine (JVM) that Content Services 9 is running on, Content Services 9 will export its MBeans to that server. Otherwise, Content Services 9 will start up its own MBean server. This means that, for example, on WebLogic, the Content Services 9 beans will compliment those provided by the application server and will be navigable in the same context with a suitable JMX client.

Activating the Sun JMX agent and local JMX connectivity

Using an application server and a Sun JVM, for the richest possible monitoring experience, you can get Content Services 9 and the application server to share the JVM's own platform MBean server, whose pre-registered MXBeans give a detailed view of the JVM's health, usage and throughput, in areas including class loading, Hotspot compilation, garbage collection and thread activity. Sun's MBean server also provides a convenient 'local' connection method, allowing the Content Services 9 process to be automatically 'discovered' by a JMX client such as JConsole without manual configuration of connection details.

The Sun JMX agent can also be activated in 'remote' mode (where a connection is made through an RMI lookup). However, since Content Services 9 is always preconfigured to allow a secure remote JMX connection on any JVM, it is most likely that you will choose to activate the Sun JMX agent in local mode.

This will mean the platform MBean Server will be shared by Content Services 9 and still be available for remote connections through the RMI connector.

- To activate the Sun JMX agent in local mode, you simply need to ensure that the following system property is set:

  ```
  com.sun.management.jmxremote
  ```

  For example, in your application server startup script, you could use the following line:

  ```
  export JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote"
  ```
- Refer to the Sun documentation for more information on all the possible configuration options.

2.4 Sizing guidelines

Accurately estimating how many servers should be used to run Content Services 9 is more of an art than it is a science. What follows is a collection of common wisdom based on practical experience and some basic assumptions. Each deployment is different and it is therefore expected that the reader apply his understanding of the requirements, use cases and preferred environment when determining the ideal configuration.

The recommendations made in this section are based on an analysis of the results of running extensive suite of benchmarks against an Content Services 9 server. Note that we will focus on Content Services 9 software only and not on dependencies such as the database or storage solutions.

This section has been organized such that it "cuts to the chase" by first offering sizing recommendations and calculations that can be immediately applied to most installations. Supporting data and additional analysis follows.

2.4.1 Assumptions

For the purposes of this discussion, we assume all servers will meet the following hardware configuration:

- **Processor:** Intel Xeon 3.16Ghz
- **Number of CPUs:** 2
- **Cores per CPU:** 4
- **Memory:** 4GB RAM
- **Disk:** 100GB (minimum)
- **JVM:** Sun Java 6 (JDK 1.6)
- **Operating System:** Windows 2008 Server

If the preferred deployment environment is not equivalent to this, care must be taken to adjust the anticipated performance figures to match the actual environment.

2.4.2 Starting production configurations

Minimal server configurations are based on whether clustering is required. Clustering is often used to improve performance and to help guarantee high-availability and fault-tolerance. This section does not discuss specific instructions for configuring a high-availability environment.

2.4.2.1 Non-clustered

In this configuration, there is one server running Content Services 9 while another runs the database. Content Services 9 and the database can co-exist on development environments and small deployments.

2.4.2.2 Clustered

In this configuration, two or more Content Services 9 servers share a common database and file-system, each on its own dedicated server for a total minimum of four servers, two running Content Services 9, one running the database, while the fourth serves as the shared fileserver.

2.4.3 Basic Sizing Methodology

2.4.3.1 Utilization Assumptions

The Content Services 9 benchmark suite is a multi-threaded utility that utilizes "virtual users" to simulate actual users accessing Contentspace 9. Every virtual user in the suite executed around 120 activities within the repository. Each activity was followed by a 5-second "think time" to more closely simulate real users. The sequence of activities was repeated 40 times by each virtual user. The benchmark suite was then executed with 40, 80, 120, 160 and 200 virtual users.

We assume that in realistic situations, utilization of the environment is heavily weighted towards READ operations while CREATE and UPDATE operations consume about 1/7th of all activities. That is to say that there is, on average, one CREATE or UPDATE operation for every seven (7) READ operations. Within Content Services 9, CREATE and UPDATE operations are the most expensive tasks.

2.4.3.2 Acceptable Performance

The goal is to provide sizing guidance with the aim to ensure that users experience reasonable performance in their day-to-day activities. In this case, we believe that, on a heavily utilized system, READ operations should, on average, execute in approximately two (2) seconds while CREATE and UPDATE operations should complete in approximately seven (7) seconds. A margin of three (3) seconds is allowed.

2.4.3.3 Sizing Formula

After reviewing the benchmark test results, we conclude that the maximum load supported by the test environment while ensuring adequate response times to be between 40 to 80 "concurrent" simulated users. For the purposes of this document, we'll say 60.

Even with the "think time" that was included in the testing suite, we believe that the "real-world" characteristics of an Content Services 9 server are such that over 100 "concurrent" users can be easily supported, but this is where the "art" of sizing comes into play.

A thorough understanding of the intended utilization of the Content Services 9 environment is required to provide realistic sizing recommendations.

Distilling the numbers above along with the benchmark results, we can safely say that the benchmark environment supported about 20 "operations" per second, where an "operation" is defined as any CREATE, READ or UPDATE activity within the system. We obtain that number through the following calculation:

Maximum number of users: 60

Avg. number of seconds per READ Operation: 2

Avg. number of seconds per CREATE & UPDATE operation: 7

Ratio of READs to CREATEs: 7:2

Weighted avg. # of seconds for all operations: 3

Avg. number of operations per second for 60 users: 60 users / 3 secs per operation = 20

Based on this number, we can achieve a close approximation of what performance will be like for a given environment.

A simple example would be a single-CPU version of the above-mentioned hardware. Such an environment yields approximate 50% performance so the number of operations per second drops to 10.

2.4.3.4 Cluster Performance

Performance on Content Services 9 cluster scales in a linear fashion though overhead introduced by clustered caches, network latency and other factors will decrease the overall performance of each server added to the cluster. For that reason, a safe number to use is 85% though the actual number is often higher. That is to say, every server added to a cluster improves overall throughput and performance by approximately 85%.

2.4.3.5 Database to Index Ratios

With full-text indexing enabled, a newly installed repository can expect to see initial index and database growth to be as follows:

1 Doc: 10K Index: 0.5K DB

That is to say, that every new document results in 10 KB index file grown and about 512 Bytes of database growth. If full-text indexing is disabled, or the greater bulk of nodes in the repository are purely metadata (no "files" attached) then one can expect the Lucene index utilization to be drastically reduced. We estimate growth to decrease to 250 bytes or less per new document.

Additionally, as the number of documents in the repository increases, the index growth will begin to taper off as fewer net-new entries are created. After extended use, the database to index size ratio will invert to approximately 2:1 which is to say, that if the database contains 200MB of data, the Lucene index will consume approximately another 100MB.

There is also a correlation between document size and Lucene index size, but this is difficult to predict and is beyond the scope of this document. A general rule of thumb is that every megabyte of indexable content will result in 5-15K of index data. Be aware that there is no correlation between document size and database growth.

Note: The ratios will change when content models are customized and more properties are added.

2.4.3.6 Connection Pool Settings

To ensure proper function, there should be approximately 1.5 database connections for every servlet engine connection. For example:

Number of Servers: 4

Servlet Engine Thread Pool (per server): 200

JDBC Connection Pool (per server): 300

MySQL Server Connections: 1200

2.5 Enabling the review of documents created from images

PDF Generator ES2 is used to generated PDF files for Content Services 9. By default, when PDF Generator ES2 creates a document based on an image file, it produces a document in PDF/A format. It is not possible to add comments to PDF/A documents.

If you want to enable commenting on review of documents created from images, you must make some changes to the Generate PDF service.

Note: This change works only on servers running Microsoft Windows. Commenting on documents created from images is not possible on non-Windows servers.

➤ **To enable the review of documents created from images:**

1. In LiveCycle Administration Console, click **Services** > **Applications and Services** > **Service Management.**
2. Search for `GeneratePDFService` and then click the service name.
3. In the **Use Acrobat Image Conversion (Windows Only)** box, set the value to `true`.

A Appendix: JMX bean categories

This reference section provides detailed information on the individual bean types exported by Content Services 9. The heading for each bean type provides the JMX object naming scheme, where possible. Each section lists the individual properties for the bean type.

A.1 JMX read-only monitoring beans

Alfresco:Name=Authority

Exposes key metrics relating to the authority service:

NumberOfGroups

The number of groups known to the Authority Service.

NumberOfUsers

The number of users known to the Authority Service.

Alfresco:Name=ConnectionPool

Allows monitoring of the Apache Commons DBCP database connection pool and its configuration. It exposes the following properties:

DefaultTransactionIsolation

The JDBC code number for the transaction isolation level, corresponding to those in the `java.sql.Connection` class. The special value of -1 indicates that the database's default transaction isolation level is in use and this is the most common setting. For the Microsoft SQL Server JDBC driver, the special value of 4096 indicates snapshot isolation.

DriverClassName

The fully-qualified name of the JDBC driver class.

InitialSize

The number of connections opened when the pool is initialized.

MaxActive

The maximum number of connections in the pool.

MaxIdle

The maximum number of connections that are not in use kept open.

MaxWait

The maximum number of milliseconds to wait for a connection to be returned before throwing an exception (when connections are unavailable) or -1 to wait indefinitely.

MinEvictableIdleTimeMillis

The minimum number of milliseconds that a connection may sit idle before it is eligible for eviction.

MinIdle

The minimum number of connections in the pool.

NumActive

The number connections in use; a useful monitoring metric.

NumIdle

The number of connections that are not in use; another useful monitoring metric.

Url

The JDBC URL to the database connection.

Username

The name used to authenticate with the database.

RemoveAbandoned

A Boolean that when true indicates that a connection is considered abandoned and eligible for removal if it has been idle longer than the RemoveAbandonedTimeout.

RemoveAbandonedTimeout

The time in seconds before an abandoned connection can be removed.

TestOnBorrow

A boolean that when true indicates that connections will be validated before being borrowed from the pool.

TestOnReturn

A boolean that when true indicates that connections will be validated before being returned to the pool.

TestWhileIdle

A Boolean that when true indicates that connections will be validated whilst they are idle.

TimeBetweenEvictionRunsMillis

The number of milliseconds to sleep between eviction runs, when greater than zero.

ValidationQuery

The SQL query that will be used to validate connections before returning them.

Alfresco:Name=ContentStore,Type=*,Root=*

Allows monitoring of each of Content Services 9 content stores. When `Type=FileContentStore`, the Root attribute of the name holds the file system path to the store. The following properties are exposed:

- TotalSize: The total size in bytes.
- WriteSupported: Stated whether the store currently allow write operations.

Alfresco:Name=ContentTransformer,Type=*

Exposes key information about the transformation utilities relied upon by Content Services 9. Currently, there are two instances:

- `Alfresco:Name=ContentTransformer,Type=ImageMagick`
- `Alfresco:Name=ContentTransformer,Type=pdf2swf`

The following properties are exposed:

- `Available`: A boolean that when true indicates that the utility is actually installed correctly and was found when the Content Services 9 server started up.
- `VersionString`: The version information returned by the utility, if it was found to be available.

Alfresco:Name=DatabaseInformation

Exposes metadata about the database itself.

DatabaseMajorVersion

The database version number.

DatabaseMinorVersion

The database version number.

DatabaseProductName

The database product name.

DatabaseProductVersion

The database product version.

DriverMajorVersion

The driver version number.

DriverMinorVersion

The driver version number.

DriverName

Product name of the JDBC driver.

DriverVersion

The driver version number.

JDBCMajorVersion

The major version number of the JDBC specification supported by the driver.

JDBCMinorVersion

The minor version number of the JDBC specification supported by the driver.

StoresLowerCaseIdentifiers

StoresLowerCaseQuotedIdentifiers

StoresMixedCaseIdentifiers

StoresMixedCaseQuotedIdentifiers

StoresUpperCaseIdentifiers

StoresUpperCaseQuotedIdentifiers

URL

The JDBC URL of the database connection.

UserName

The name used to authenticate with the database.

Alfresco:Name=Hibernate

An instance of the `StatisticsService` class provided by Hibernate, allowing access to an extensive set of Hibernate-related metrics.

Alfresco:Name=LuceneIndexes,Index=*

Allows monitoring of each searchable index. The Index attribute of the name holds the relative path to the index under `alf_data/lucene-indexes` and the following properties are exposed:

ActualSize

The size of the index in bytes.

EntryStatus

A composite table containing the current status of each entry in the index (double-click the value in JConsole to expand it and view its rows). Each row in the table has a key of the format `<ENTRY TYPE>-<ENTRY STATE>`, for example, `DELTA-COMMITTED` and a value containing the number of entries with that type and state.

EventCounts

A composite table containing the names and counts of significant events that have occurred on the index since the server was started (double-click the value in JConsole to expand it and view its rows). Examples of event names are `CommittedTransactions, MergedDeletions,` and `MergedIndexes`.

NumberOfDocuments

The number of documents in the index.

NumberOfFields

The number of fields known to the index.

NumberOfIndexedFields

The number of these fields that are indexed.

UsedSize

The size of the index directory in bytes. A large discrepancy from the value of ActualSize may indicate that there are unused data files.

Alfresco:Name=ModuleService

Allows monitoring of installed modules.

AllModules

A composite table containing the details of all modules currently installed. Double-click the value in JConsole to expand it and use the Composite Navigation arrows to navigate through each module.

Alfresco:Name=OpenOffice

Exposes information about the OpenOffice server used for document conversions. In addition to the property below, this bean has a property corresponding to each registry key in the sub-tree of the OpenOffice configuration registry, providing useful metadata about the particular flavor of OpenOffice that is installed. For example, ooName provides the product name, for example, "`OpenOffice.org`" and `ooSetupVersionAboutBox` provides its version, for example, "3.0.0".

available

A Boolean that when true indicates that a connection was successfully established to the OpenOffice server.

Alfresco:Name=PatchService

Allows monitoring of installed patches.

AppliedPatches

A composite table containing the details of all patches currently installed. Double-click the value in JConsole to expand it and use the "Composite Navigation" arrows to navigate through each patch.

Alfresco:Name=RepositoryDescriptor,Type=*

Exposes meta data about the Content Services 9 repository. Currently, there are two instances of this bean:

Alfresco:Name=RepositoryDescriptor,Type=Installed

Exposes information about the initial repository installation, before any patches or upgrades were installed. Of most relevance to patch and upgrade scenarios.

Alfresco:Name=RepositoryDescriptor,Type=Server

Exposes information about the current server version, as contained in the Content Services 9 war file. This instance should be used to determine the current properties of the server.

Both expose the following properties:

Edition: The Content Services 9 edition, for example, "Enterprise".

Id: The repository unique ID. This property is only available from the Installed descriptor.

Name

The repository name.

Schema

The schema version number.

Version

The full version string, including build number, for example, "3.1.0 (stable r1234)".

VersionBuild

The build number.

VersionLabel

An optional label given to the build, such as "dev" or "stable".

VersionMajor

The first component of the version number.

VersionMinor

The second component of the version number.

VersionNumber

The full version number, composed from major, minor and revision numbers.

VersionRevision

The third component of the version number.

Alfresco:Name=Runtime

Exposes basic properties about the memory available to the JVM. Note that a Sun JVM exposes much more detailed information through its platform MX Beans.

FreeMemory

The amount of free memory in bytes.

MaxMemory

The maximum amount of memory that the JVM will attempt to use in bytes.

TotalMemory

The total amount of memory in use in bytes.

Alfresco:Name=Schedule,Group=*,Type=*,Trigger=*

Allows monitoring of the individual triggers, i.e. scheduled jobs, running in the Quartz scheduler. The attributes of the object name have the following meaning:

Group

The name of the schedule group that owns the trigger. Typically DEFAULT.

Type

The type of trigger, typically MonitoredCronTrigger or MonitoredSimpleTrigger. Triggers of different types have different properties, as you will see below.

Trigger

The name of the trigger itself. Must be unique within the group.

All instances have the following properties:

CalendarName

The name of the scheduling Calendar associated with the trigger, or null if there is not one.

Description

An optional textual description of the trigger.

EndTime

The time after which the trigger will stop repeating, if set.

FinalFireTime

The time at which the last execution of the trigger is scheduled, if applicable.

Group

The name of the schedule group that owns the trigger.

JobGroup

The name of the schedule group that owns the job executed by the trigger.

JobName

The name of the job executed by the trigger.

MayFireAgain

A Boolean that when true indicates that it is possible for the trigger to fire again.

Name

The name of the trigger.

NextFireTime

The next time at which the trigger will fire.

PreviousFireTime

The previous time at which the trigger fired.

Priority

A numeric priority that decides which trigger is executed before another in the event of a 'tie' in their scheduled times.

StartTime

The time at which the trigger should start.

State

The current state of the trigger.

Volatile

A Boolean that when true indicates that the trigger will not be remembered when the JVM is restarted.

When `Type=MonitoredCronTrigger`, the following additional properties are available:

CronExpression

A UNIX-like expression, using the same syntax as the cron command, that expresses when the job should be scheduled.

TimeZone

The name of the time zone to be used to interpret times.

When `Type=MonitoredSimpleTrigger` the following additional properties are available:

RepeatCount

The number of times the job should repeat, after which it will be removed from the schedule. A value of -1 means repeat indefinitely.

RepeatInterval

The time interval in milliseconds between job executions.

TimesTriggered

The number of times the job has been run.

Alfresco:Name=SystemProperties

A dynamic MBean exposing all the system properties of the JVM. The set of standard system properties is documented on the Apache website.

A.2 JMX configuration beans

This section contains the list of configuration beans. Content Services 9 introduces an innovative way to manage the configuration of the individual Spring beans that compose the server. This feature is available for security and authentication configuration, which can be particularly complex to manage given the possibility of multiple-chained authentication services and authentication components, each with their own DAOs and other supporting services.

To help with the management of such configuration, the key properties of key authentication bean classes are annotated with a special @Managed annotation, that causes them to be exposed automatically through dynamic MBeans under the `Alfresco:Type=Configuration` naming tree. This means that the key beans that make up your authentication chain will become visible to a JMX client, no matter how they are named and wired together.

The current set of authentication classes that have this facility include:

- Authentication Components, including chained, JAAS, LDAP and NTLM components
- Authentication Services, including chained and unchained
- Authentication DAOs
- `LDAPInitialDirContextFactories`, encapsulating the parameters of the LDAP server
- `LDAPPersonExportSource`, controlling the synchronization of person information with an LDAP server

In JConsole, the view of a server with a particularly complex authentication configuration that shows all the authentication classes are visible under the `Alfresco:Type=Configuration` naming tree and navigable with JConsole. These beans provide a read-only view of the configuration.

A.3 JMX editable management beans

This section contains the list of editable management beans.

Alfresco:Name=FileServerConfig

Allows management and monitoring of the various file servers.

A.3.1 Read-only properties

CIFSServerAddress

Not implemented.

CIFSServerName

The CIFS server name, if available.

A.3.2 Editable Properties

Note: These are not cluster-aware. If more than one file server is running (for example, load-balanced FTP) then changes will need to be applied to each machine. Some consoles (for example, JManage) may provide basic facilities for accessing each machine in an application cluster.

CIFSServerEnabled

A Boolean that when true indicates that the CIFS server is enabled and functioning.

FTPServerEnabled

A Boolean that when true indicates that the FTP server is enabled and functioning.

NFSServerEnabled

A Boolean that when true indicates that the NFS server is enabled and functioning.

Alfresco:Name=Log4jHierarchy

An instance of the `HierarchyDynamicMBean` class provided with log4j that allows adjustments to be made to the level of detail included in the Content Services 9 server's logs. Note that it is possible to run Content Services 9 using JDK logging instead of log4j, in which case this bean will not be available.

A.3.3 Read-only properties

The bean has a property for each logger known to `log4j`, whose name is the logger name, usually corresponding to a Java class or package name, and whose value is the object name of another MBean that allows management of that logger (see `#log4j:logger=*)`. Despite how it might seem, these properties are read-only and editing them has no effect.

A.3.4 Editable properties

There is one special editable property and note again that it is not cluster aware.

threshold

Controls the server-wide logging threshold. Its value must be the name of one of the log4j logging levels. Any messages logged with a priority lower than this threshold will be filtered from the logs. The default value is ALL, which means no messages are filtered, and the highest level of filtering is OFF which turns off logging altogether (not recommended).

A.3.5 Operations with Impact

addLoggerMBean

This adds an additional logger to the hierarchy, meaning that the bean will be given an additional read-only property for that logger and a new MBean will be registered in the `#log4j:logger=* tree`, allowing management of that logger. Is is not normally necessary to use this operation, because the Content Services 9 server pre-registers all loggers initialized during startup. However, there may be a chance that the logger you are interested in was not initialized at this point, in which case you will have to use this operation. The operation requires the fully qualified name of the logger as an argument and if successful returns the object name of the newly registered MBean for managing that logger.

For example, if in Java class `org.alfresco.repo.admin.patch.PatchExecuter` the logger is initialized as follows:

```
private static Log logger = LogFactory.getLog(PatchExecuter.class);
```

Then the logger name would be `org.alfresco.repo.admin.patch.PatchExecuter`.

log4j:logger=*

An instance of the LoggerDynamicMBean class provided with log4j that allows adjustments to be made to the level of detail included in the logs from an individual logger. Note that it is possible to run Content Services 9 using JDK logging instead of log4j, in which case this bean will not be available.

A.3.6 Read-only properties

name

The logger name

A.3.7 Editable properties

There is one special editable property and note again that it is not cluster aware.

priority

The name of the minimum log4j logging level of messages from this logger to include in the logs. For example, a value of ERROR would mean that messages logged at lower levels such as WARN and INFO would not be included.

Alfresco:Name=RepoServerMgmt

Provides monitoring and management capabilities for low-level repository functions. The managed properties and operations are cluster-aware, that is, changes will be replicated across all nodes.

TicketCountAll

Count of all authentication tickets, including those expired.

TicketCountNonExpired

Count of unexpired authentication tickets.

UserCountAll

Count of all users who have logged in, including those with expired authentication tickets.

UserCountNonExpired

Count of all logged-in users with unexpired authentication tickets.

A.3.8 Editable properties

LinkValidationDisabled

A Boolean that when true indicates that the link validation service is enabled.

MaxUsers

The limit for non-expired user logins: -1 if no limit set, 0 to prevent further logins.

ReadOnly

A Boolean that when true indicates that the repository is in read only mode, that is, no write operations are possible.

SingleUserOnly

Set to the name of a single user, for example, "admin", to only allow access by that single user or leave blank to allow access by all users.

A.3.9 Operations with impact

invalidateTicketsAll

Invalidate all tickets, expired and unexpired.

invalidateTicketsExpired

Invalidate expired tickets only.

invalidateUser

Invalidate all tickets for the given user name.

A.3.10 Operations with no impact

listUserNamesAll

Gets the names of all users who have logged in, including those with expired authentication tickets.

listUserNamesNonExpired

Gets the names of all logged-in users with unexpired authentication tickets.

Alfresco:Name=VirtServerRegistry,Type=VirtServerRegistry

This is used directly by the LiveCycle ES2 Virtualization Server.