CoreMedia Digital Experience Platform 8
//Version 7.5.45-10

CoreMedia IBM Deployment Manual

CoreMedia IBM Deployment Manual

CoreMedia AG

Ludwig-Erhard-Straße 18

20459 Hamburg

International

Germany

Licenses and Trademarks

07.Mar 2017

List of Figures

List of Tables

List of Examples

1. Preface

This manual describes how you create and deploy CoreMedia artifacts for and into IBM WebSphere and how you configure the IBM systems for use with CoreMedia components.

- Chapter 3, *Configuring DB2 for Use With the Content Server* [14] describes how to set up a DB2 database for use with the CoreMedia *content server*.
- Chapter 4, *Creating Deployment Packages* [16] describes how to use the *CoreMedia Blueprint* workspace in order to create WAR artifacts that can be deployed into IBM WebSphere.
- Chapter 5, *Settings for IBM WebSphere Application Server* [36] describes how to prepare the IBM WebSphere Application Server for CoreMedia system deployment.
- Chapter 7, *Known Limitations* [44] describes some known limitations for deployment of *CoreMedia Digital Experience Platform 8* into IBM WebSphere.

1.1 Audience

This manual is intended for developers who create CoreMedia applications for deployment into IBM WebSphere and for administrators who want to deploy CoreMedia components into IBM WebSphere Application Server. The reader should be familiar with the IBM WebSphere Application Server and with the CoreMedia Blueprint workspace concepts and usage.

1.2 Typographic Conventions

CoreMedia uses different fonts and types in order to label different elements. The following table lists typographic conventions for this documentation:

Table 1.1. Typographic conventions

Element	**Typographic format**	**Example**
Source code Command line entries Parameter and values Class and method names Packages and modules	Courier new	`cm systeminfo start`
Menu names and entries	Bold, linked with |	Open the menu entry **Format|Normal**
Field names CoreMedia Components Applications	Italic	Enter in the field *Heading* The *CoreMedia Component* Use *Chef*
Entries	In quotation marks	Enter "On"
(Simultaneously) pressed keys	Bracketed in "<>", linked with "+"	Press the keys <Ctrl>+<A>
Emphasis	Italic	It is *not* saved
Buttons	Bold, with square brackets	Click on the **[OK]** button

Code lines in code examples which continue in the next line	\	`cm systeminfo \`
		`-u user`

Mention of other manuals	Square Brackets	See the [Studio Developer Manual] for more information.

In addition, these symbols can mark single paragraphs:

Table 1.2. Pictographs

Pictograph	**Description**
ℹ	Tip: This denotes a best practice or a recommendation.
⚠	Warning: Please pay special attention to the text.

Pictograph	Description
⚡	Danger: The violation of these rules causes severe damage.

1.3 CoreMedia Services

This section describes the CoreMedia services that support you in running a CoreMedia system successfully. You will find all the URLs that guide you to the right places. For most of the services you need a CoreMedia account. See Section 1.3.1, “Registration” [5] for details on how to register.

CoreMedia User Orientation for CoreMedia Developers and Partners

Find the latest overview of all CoreMedia services and further references at:

http://documentation.coremedia.com/new-user-orientation

- Section 1.3.1, “Registration” [5] describes how to register for the usage of the services.
- Section 1.3.2, “CoreMedia Releases” [5] describes where to find the download of the software.
- Section 1.3.3, “Documentation” [6] describes the CoreMedia documentation. This includes an overview of the manuals and the URL where to find the documentation.
- Section 1.3.4, “CoreMedia Training” [8] describes CoreMedia training. This includes the training calendar,the curriculum and certification information.
- Section 1.3.5, “CoreMedia Support” [9] describes the CoreMedia support.

1.3.1 Registration

In order to use CoreMedia services you need to register. Please, start your initial registration via the CoreMedia website. Afterwards, contact the CoreMedia Support (see Section 1.3.5, “CoreMedia Support” [9]) by email to request further access depending on your customer, partner or freelancer status so that you can use the CoreMedia services.

1.3.2 CoreMedia Releases

Downloading and Upgrading the Blueprint Workspace

CoreMedia provides its software as a Maven based workspace. You can download the current workspace or older releases via the following URL:

http://releases.coremedia.com/dxp8

Refer to our Blueprint Github mirror repository for recommendations to upgrade the workspace either via Git or patch files.

> If you encounter a 404 error then you are probably not logged in at GitHub or do not have sufficient permissions yet. See Section 1.3.1, “Registration” [5] for details about the registration process. If the problems persist, try clearing your browser cache and cookies.

Maven artifacts

CoreMedia provides its release artifacts via Maven under the following URL:

https://repository.coremedia.com

You have to add your CoreMedia credentials to your Maven settings file as described in section CoreMedia Digital Experience Platform 8 Developer Manual.

License files

You need license files to run the CoreMedia system. Contact the support (see Section 1.3.5, “CoreMedia Support” [9]) to get your licences.

1.3.3 Documentation

CoreMedia provides extensive manuals and Javadoc as PDF files and as online documentation at the following URL:

http://documentation.coremedia.com/dxp8

The manuals have the following content and use cases:

Table 1.3. CoreMedia manuals

Manual	Audience	Content
CoreMedia Utilized Open-Source Software	Developers, architects, administrators	This manual lists the third-party software used by CoreMedia and lists, when required, the licence texts.
Supported Environments	Developers, architects, administrators	This document lists the third-party environments with which you can use the CoreMedia system, Java versions or operation systems for example.
Studio User Manual, English	Editors	This manual describes the usage of *CoreMedia Studio* for editorial and administrative work. It also describes the usage of the *Adaptive Personalization* and *Elastic Social* GUI that are integrated into *Studio*.

Manual	Audience	Content
LiveContext for IBM WebSphere Manual	Developers, architects, administrators	This manual gives an overview over the structure and features of CoreMedia LiveContext. It describes the integration with the IBM WebSphere Commerce system, the content type model, the *Studio* extensions, folder and user rights concept and many more details. It also describes administrative tasks for the features. It also describes the concepts and usage of the project workspace in which you develop your CoreMedia extensions. You will find a description of the Maven structure, the virtualization concept, learn how to perform a release and many more.
Operations Basics Manual	Developers, administrators	This manual describes some overall concepts such as the communication between the components, how to set up secure connections, how to start application or the usage of the watchdog component.
Adaptive Personalization Manual	Developers, architects, administrators	This manual describes the configuration of and development with *Adaptive Personalization*, the CoreMedia module for personalized websites. You will learn how to configure the GUI used in *CoreMedia Studio*, how to use predefined contexts and how to develop your own extensions.
Analytics Connectors Manual	Developers, architects, administrators	This manual describes how you can connect your CoreMedia website with external analytic services, such as Google Analytics.
Content Application Developer Manual	Developers, architects	This manual describes concepts and development of the *Content Application Engine (CAE)*. You will learn how to write JSP or Freemarker templates that access the other CoreMedia modules and use the sophisticated caching mechanisms of the CAE.
Content Server Manual	Developers, architects, administrators	This manual describes the concepts and administration of the main CoreMedia component, the *Content Server*. You will learn about the content type model which lies at the heart of a CoreMedia system, about user and rights management, database configuration, and more.

Manual	**Audience**	**Content**
Elastic Social Manual	Developers, architects, administrators	This manual describes the concepts and administration of the *Elastic Social* module and how you can integrate it into your websites.
Importer Manual	Developers, architects	This manual describes the structure of the internal CoreMedia XML format used for storing data, how you set up an *Importer* application and how you define the transformations that convert your content into CoreMedia content.
Search Manual	Developers, architects, administrators	This manual describes the configuration and customization of the *CoreMedia Search Engine* and the two feeder applications: the *Content Feeder* and the *CAE Feeder*.
Site Manager Developer Manual	Developers, architects, administrators	This manual describes the configuration and customization of *Site Manager*, the Java based stand-alone application for administrative tasks. You will learn how to configure the *Site Manager* with property files and XML files and how to develop your own extensions using the *Site Manager API*.
Studio Developer Manual	Developers, architects	This manual describes the concepts and extension of *CoreMedia Studio*. You will learn about the underlying concepts, how to use the development environment and how to customize *Studio* to your needs.
Unified API Developer Manual	Developers, architects	This manual describes the concepts and usage of the *CoreMedia Unified API*, which is the recommended API for most applications. This includes access to the content repository, the workflow repository and the user repository.
Workflow Manual	Developers, architects, administrators	This manual describes the *Workflow Server*. This includes the administration of the server, the development of workflows using the XML language and the development of extensions.

If you have comments or questions about CoreMedia's manuals, contact the Documentation department:

Email: documentation@coremedia.com

1.3.4 CoreMedia Training

CoreMedia's training department provides you with the training for your CoreMedia projects either in the CoreMedia training center or at your own location.

You will find information about the CoreMedia training program, the training schedule and the CoreMedia certification program at the following URL:

http://www.coremedia.com/training

Contact the Training department at the following email address:

Email: training@coremedia.com

1.3.5 CoreMedia Support

CoreMedia's support is located in Hamburg and accepts your support requests between 9 am and 6 pm MET. If you have subscribed to 24/7 support, you can always reach the support using the phone number provided to you.

To submit a support ticket, track your submitted tickets or receive access to our forums visit the CoreMedia Online Support at:

http://support.coremedia.com/

Do not forget to request further access via email after your initial registration as described in Section 1.3.1, "Registration" [5]. The support email address is:

Email: support@coremedia.com

Create a support request

Support request

CoreMedia systems are distributed systems that have a rather complex structure. This includes, for example, databases, hardware, operating systems, drivers, virtual machines, class libraries and customized code in many different combinations. That's why CoreMedia needs detailed information about the environment for a support case. In order to track down your problem, provide the following information:

- Which CoreMedia component(s) did the problem occur with (include the release number)?
- Which database is in use (version, drivers)?
- Which operating system(s) is/are in use?
- Which Java environment is in use?
- Which customizations have been implemented?
- A full description of the problem (as detailed as possible)
- Can the error be reproduced? If yes, give a description please.
- How are the security settings (firewall)?

In addition, log files are the most valuable source of information.

To put it in a nutshell, CoreMedia needs:

Support checklist

1. a person in charge (ideally, the CoreMedia system administrator)
2. extensive and sufficient system specifications
3. detailed error description
4. log files for the affected component(s)
5. if required, system files

Log files

An essential feature for the CoreMedia system administration is the output log of Java processes and CoreMedia components. They're often the only source of information for error tracking and solving. All protocolling services should run at the highest log level that is possible in the system context. For a fast breakdown, you should be logging at debug level. The location where component log output is written is specified in its < `appName>-logback.xml` file.

Which Log File?

Mostly at least two CoreMedia components are involved in errors. In most cases, the *Content Server* log files in `coremedia.log` files together with the log file from the client. If you are able locate the problem exactly, solving the problem becomes much easier.

Where do I Find the Log Files?

By default, log files can be found in the CoreMedia component's installation directory in `/var/logs` or for web applications in the `logs/` directory of the servlet container . See the "Logging" chapter of the [Operations Basics Manual] for details.

Table 1.4. Log files check list

Component	Problem	Log files
CoreMedia Studio	general	`CoreMedia-Studio.log` `coremedia.log`

CoreMedia Editor	general	`editor.log`
		`coremedia.log`
		`workflowserver.log`
		`capclient.properties`

	check-in/check-out	`editor.log`
		`coremedia.log`
		`workflowserver.log`
		`capclient.properties`

	publication or preview	`coremedia.log` (*Content Management Server*) `coremedia.log` (Master Live Server)

Component	Problem	Log files
		`workflowserver.log` `capclient.properties`

	import	`importer.log`
		`coremedia.log`
		`capclient.properties`

	workflow	`editor.log`
		`workflow.log`
		`coremedia.log`
		`capclient.properties`

	spell check	`editor.log`
		`MS Office version details`
		`coremedia.log`

	licenses	`coremedia.log` (*Content Management Server*) `coremedia.log` (Master Live Server)

Server and client	communication errors	`editor.log`
		`coremedia.log`
		(*Content Management Server*)
		`coremedia.log`
		(Master Live Server)
		`*.jpif files`

	preview not running	`coremedia.log` (content server) `preview.log`

	website not running	`coremedia.log`
		(*Content Management Server*)
		`coremedia.log`
		(Master Live Server)
		coremedia.log
		(*Replication Live Server*)
		`Blueprint.log`
		`capclient.properties`
		`license.zip`

Server	not starting	`coremedia.log`
		(*Content Management Server*)
		`coremedia.log`
		(Master Live Server)
		`coremedia.log`
		(*Replication Live Server*)
		`capclient.properties`
		`license.zip`

1.4 Change Chapter

In this chapter you will find a table with all major changes made in this manual.

Table 1.5. Changes

Section	Version	Description
Chapter 6, *Deploying CoreMedia Collaborative Components without MongoDB* [43]	7.5.35	For deploying collaborative features without MongoDB, an in-memory setup is now recommended.
Section “Studio” [30]	7.5.41	Removed property `controlroom.jdbc.url`. *Control Room* no longer supports *IBM DB2* for persisting collaboration data. See [CoreMedia DXP 8 Manual], Section "In-Memory Replacement for MongoDB-Based Services".

2. Overview

All CoreMedia components can be deployed into the IBM technology stack that consists of the IBM WebSphere application server, AIX operating system (support planned for AEP2), and the IBM DB2 database. Before deploying to the IBM platform, the CoreMedia workspace components have to be preconfigured for this type of deployment. This manual outlines the necessary steps for a successful deployment. Consult the Supported Environments document for the specific versions of supported IBM technology.

The deployment of *CoreMedia Digital Experience Platform 8* into IBM WebSphere Commerce is described in the CoreMedia Digital Experience Platform 8 Developer Manual.

3. Configuring DB2 for Use With the Content Server

CoreMedia *Content Servers* can use IBM DB2 databases for their repositories. This section describes DB2 specific settings. For general advice see Section 3.3, “Configuring the Database” in *CoreMedia Content Server Manual*.

General Notes

The *CoreMedia Content Server* database configuration requires one DB2 user per content server. Each user will own one schema in the given database.

JDBC Configuration

Each DB2 installation comes with its own JDBC drivers, which you will find below `sqllib/java`. Ensure that `db2jcc.jar` is added to your classpath (for example, place it into the `lib/` folder of the *CoreMedia Content Server*).

Configure the `sql.properties` as follows:

Example 3.1. sql.properties Template for IBM DB2

```
sql.store.driver=com.ibm.db2.jcc.DB2Driver
sql.store.url=jdbc:db2://<DB-HOST>:<DB-PORT>/<DB-NAME>
sql.store.user=<DB-USER>
sql.store.password=<DB-USER-PASSWORD>
```

Database Configuration

The recommended configuration for the DB2 database is as follows:

- the *default buffer pool and table space page size* should be set to 32K,
- choose code set UTF-8,
- for large content types, increase the page size for the default tablespace and
- in case of problems with long queries adjust the parameters `applheapsz`, `stmtheap` and `appl_memory`.

Suggested values can be found in the example script below.

Sample Script for Database Setup

The following script is an example of how to configure the database "CM" for use with the *CoreMedia Content Server*. The script relies on the following preconditions:

- you are logged in as database system administrator and
- you have created a system user called `cmowner` (configurable through `DBOWNER_USER`).

Example 3.2. Example for Bash Script for IBM DB2 Database Setup

```
#!/bin/bash
DBNAME=CM
DBCODESET=UTF-8
DBOWNER_USER=cmowner

db2 create db ${DBNAME} \
  using codeset ${DBCODESET} \
  territory us collate using system \
  pagesize 32768

db2 update db cfg for ${DBNAME} using applheapsz 16384
db2 update db cfg for ${DBNAME} using stmtheap 60000
db2 update db cfg for ${DBNAME} using logfilsiz 131072
db2 update db cfg for ${DBNAME} using maxlocks 100
db2 update db cfg for ${DBNAME} using locklist 65536

db2 connect to ${DBNAME}
db2 create bufferpool bp_${DBNAME} \
  all dbpartitionnums size 50000 numblockpages 0 pagesize 32768
db2 create tablespace ts_${DBNAME} pagesize 32768 \
  managed by automatic storage \
  extentsize 32 prefetchsize 64 bufferpool bp_${DBNAME}
db2 create system temporary tablespace tts_${DBNAME} \
  pagesize 32768 managed by automatic storage \
  extentsize 32 prefetchsize 64 bufferpool bp_${DBNAME}
db2 grant use of tablespace ts_${DBNAME} to PUBLIC
# ensures that ts_${DBNAME} becomes the default tablespace
db2 drop tablespace userspace1

db2 grant CONNECT,DBADM,BINDADD,CREATE_NOT_FENCED_ROUTINE,\
  IMPLICIT_SCHEMA,LOAD,CREATE_EXTERNAL_ROUTINE,\
  QUIESCE_CONNECT,CREATETAB on database to user ${DBOWNER_USER}

db2 disconnect current
db2 detach
```

4. Creating Deployment Packages

CoreMedia Blueprint is geared towards fast development and deployment round-trips. Most of CoreMedia's customers leverage open source technologies such as Tomcat on the local developers’ workstations, independent of the final deployment platform. To support an efficient development round-trip from within your IDE, the CoreMedia web applications are preconfigured to run in a Tomcat instance started by Maven using the *tomcat7-maven-plugin*. To create deployable packages for IBM WebSphere, the standard workspace artifacts need to be modified for the specific IBM WebSphere deployment environments.

By default, the *CoreMedia Blueprint* workspace supports a deployment process using RPMs for RHEL-based Linux distributions (for Microsoft Windows and other non RPM-based operating systems, a ZIP file based approach is provided). The workspace creates these deployment artifacts automatically within the Maven module hierarchy below the `packages` folder in the workspace.

Before these deployment packages are created, the applications themselves are packaged as standard web archives (WAR) and are ready for deployment within any Enterprise Java application server, including IBM WebSphere Application Server. The web archives are created by Maven modules below the `modules` folder in the workspace. The Maven modules that produce WAR archives are named `*-webapp`, for example, `modules/studio/studio-webapp`. For WAR file based deployment, the creation of artifacts underneath the `packages` folder, except for the *Site Manager* module which is located at `packages/editor-webstart-webapp`, can be skipped.

In order to deploy these artifacts to an application server, a new configuration layer needs to be created for your specific target environments. This includes the reconfiguration or replacement of local development configuration files with environment-specific versions. For web applications, you might need to exclude third-party libraries that conflict with the WebSphere Application Server from the web application's `lib` directory. Also do these exclusions in the extra configuration layer.

4.1 Defining the Build Process

Configuring certain properties or excluding dependencies directly in the source directory would disable IDE support for local development. To keep this support, there are several approaches for creating deployable artifacts without changing any of those configuration files directly, as outlined below.

Independent of the approach you have chosen, all web application modules have to be built with the `websphere` profile being activated.

4.1.1 Manually Modifying Artifacts

One approach for configuring properties is to manually change the web application archive. To do so, extract the archive, modify the configuration files or add new ones and delete libraries from the `WEB-INF/lib` folder before you archive again, using a standard ZIP utility.

To help you locate the packages, the following table maps applications to their artifact paths in the workspace.

Table 4.1. Artifact Paths in the Workspace

Content Management Server	`modules/server/content-management-server-webapp/target/coremedia.war`
Master Live Server	`modules/server/master-live-server-webapp/target/coremedia.war`
Replication Live Server	`modules/server/replication-live-server-webapp/target/coremedia.war`
User Changes	`modules/server/user-changes-webapp/target/user-changes.war`
Workflow Server	`modules/server/workflow-server-webapp/target/workflow.war`
Solr Web application	Get it from your local Maven repository or modify and archive `modules/search/solr-webapp/target/solr`
Solr Home Configuration	`modules/search/solr-config/target/solr-config.zip`
Content Feeder	`modules/search/content-feeder-webapp/target/content-feeder.war`
CAE Feeder Preview	`modules/search/caefeeder-preview-webapp/target/caefeeder-preview.war`

CAE Feeder Live	modules/search/caefeeder-live-webapp/target/caefeeder-live.war
Elastic Worker	modules/elastic-worker/elastic-worker-webapp/target/elastic-worker.war
Preview	modules/cae/cae-preview-webapp/target/blueprint.war
Delivery	modules/cae/cae-live-webapp/target/blueprint.war
Studio	modules/studio/studio-webapp/target/studio.war
WebDAV	modules/editor-components/webdav-webapp/target/webdav.war
Site Manager	packages/editor-webstart-webapp/target/editor-webstart.war

4.1.2 Using the coremedia-application-maven-plugin

Another approach for configuring artifacts is to use the coremedia-application-maven-plugin. Here you would create an additional Maven module layer similar to the *maven-war-plugin* approach, but use the ability of the *coremedia-application-maven-plugin* to define only the property delta you intend to change. For a description of this approach, see Section 4.3.9, “Configure Filtering in the Workspace” in *CoreMedia Digital Experience Platform 8 Developer Manual* and the plugin documentation here.

You can also use the merge ability to define only a single Maven module for all applications, although not all use cases of the *maven-war-plugin* scenario can be mapped. Adding or excluding dependencies is not supported. However, adding libraries can be realized using IBM WebSphere shared libraries. Excluding libraries can be achieved using excludes when repackaging the WAR archives afterwards.

Example 4.1. Create Packages using coremedia-application-maven-plugin

```
...
  <dependencies>
    <dependency>
      <groupId>com.coremedia.blueprint</groupId>
      <artifactId>content-management-server-webapp</artifactId>
      <version>${project.version}</version>
      <type>war</type>
    </dependency>
    ...
  </dependencies>
  <build>
    <plugins>
      <!--
       | to modify properties within an existing property file, create a property
       | file with the properties you want to override below
       | src/main/override-properties.
       | Name the file identical to its target file and place it in a
       | path matching the path within the resulting
       | coremedia-application archive.
       | i.e. the webapps context is cms, you want to change the
       | cap.server.http.port property in the contentserver.properties file and
       | the path to the property file is WEB-INF/properties/corem.
       | Then the contentserver.properties file containing the delta must be
       | placed below src/main/override-properties/webapps/cms plus the path
       | to the properties file mentioned above.
      -->
      <plugin>
        <groupId>com.coremedia.maven</groupId>
        <artifactId>coremedia-application-maven-plugin</artifactId>
        <version>2.7.9</version>
        <extensions>true</extensions>
        <executions>
          <execution>
            <id>assemble-apps</id>
            <goals>
              <goal>package-inplace</goal>
            </goals>
            <configuration>
              <tomcatPath>/</tomcatPath>
              <webapps>
                <webapp>
                  <groupId>com.coremedia.blueprint</groupId>
                  <artifactId>content-management-server-webapp</artifactId>
                  <context>cms</context>
                </webapp>
                ...
              </webapps>
            </configuration
          </execution>
        </executions>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-antrun-plugin</artifactId>
        <version>1.7</version>
        <executions>
          <execution>
            <id>default-cli</id>
            <goals>
              <goal>run</goal>
            </goals>
            <phase>package</phase>
            <configuration>
              <target>
                <war update="true"
                     manifest="${basedir}/src/main/MANIFEST.MF"
                     compress="false"
                     destfile="target/cms.war"
                     basedir="target/${project.build.finalName}/webapps/cms"
                     needxmlfile="false"/>
              </target>
            </configuration>
           </execution>
        </executions>
      </plugin>
      ...
    </plugins>
  </build>
  ...
```

4.1.3 Using the maven-war-plugin

This approach involves creating an additional Maven module layer, with one extra Maven module of packaging type `war` for each web application. In each of these extra modules, add a dependency to the original web application artifact.

To adjust configuration files with this approach, the files must be duplicated and maintained like its original. For web applications which support the CoreMedia component approach, it is sufficient to duplicate the `WEB-INF/application.properties` and add or change properties only within this file. The component approach ensures that properties listed in this file override any other occurrences in other property files.

Libraries that need to be excluded for IBM WebSphere deployment, can only be excluded using the *maven-war-plugin* and its `packagingExcludes` configuration element. For a complete description visit the documentation here.

Example 4.2. Exclude JEE libraries provided by IBM WebSphere

```
...
  <dependencies>
    <dependency>
      <groupId>com.coremedia.blueprint</groupId>
      <artifactId>content-management-server-webapp</artifactId>
      <version>${project.version}</version>
      <type>war</type>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      ...
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.6</version>
        <configuration>
          <packagingExcludes>WEB-INF/lib/foo-bar-*.jar</packagingExcludes>
        </configuration>
      </plugin>
      ...
    </plugins>
  <build>
  ...
```

To add libraries, simply add dependencies to the module. An alternative to this method is shared libraries within IBM WebSphere Application Server.

4.1.4 Recommended Process

It is recommended to use the *maven-war-plugin* approach described in Section 4.1.3, “Using the maven-war-plugin” [20], as it is the most flexible one. The disadvantage of duplicating the properties and configuration files is outweighed by several other advantages:

- The *maven-war-plugin* produces an archive as the resulting artifact, there is no need to configure additional steps. Because the resulting `WAR` file is automatically the main artifact of the module, it can be deployed to a repository using Mavens `deploy` goal. With the *coremedia-application-maven-plugin*, you would have to use an extra plugin to attach the artifact to the module before you can deploy it with Maven.

- The fact that each web application is mapped to one Maven module allows individual versioning of the deployable artifacts. The idea is not to use the *maven-release-plugin* at all, but to keep stable snapshot versions for all modules below the `modules` folder and use fix versioned POM files for all the new WebSphere deployment modules. In combination with a VCS that has a unique ID for each state of its repository; this ID is the only important reference to the source state of your deployed application. To identify the state of libraries in production, make sure that these IDs are written to the manifest `META-INF/MANIFEST.MF` of any library you create in the workspace.
- Overwriting property and configuration files instead of filtering or modifying them has the advantage that file attributes like creation time or last modification date remain the same when the original data has not changed. Recreating or redeploying an unchanged archive can be detected and skipped.

4.2 Configure the Build Process

Similar to building RPMs, the build process for WebSphere deployments can be separated into preconfiguration and post-configuration. Manually modifying `WAR` files is a form of post-configuration, whereas the other approaches mentioned before are preconfigured ones.

To help you with configuring the applications for IBM WebSphere Application Server deployment, the following tables list the minimal set of properties you need to modify to wire the CoreMedia web applications. Remember, that except for WebDAV, Solr and Site Manager, all web applications support the component approach and can be configured solely by modifying the `WEB-INF/application.properties`. The original property file names and their paths are only listed here as a reference to find the origin and possible documentation in form of comments.

In the example values listed in the tables of the following sections about application specific configurations, assumptions were made over the context paths the web applications are deployed to. However, they are completely arbitrary and you can choose them yourself. For the example the following context paths to web application mapping were chosen.

Table 4.2. Context Paths

Content Management Server	cms
Master Live Server	mls
Replication Live Server	rls
Workflow Server	wfs
Content Feeder	contentfeeder
User Changes	user-changes
Elastic Worker	elastic-worker
Solr Search Engine (Master)	solr
CAE Feeder Preview	caefeeder-preview
CAE Feeder Preview	caefeeder-live
Preview CAE	preview
Live CAE	delivery
WebDAV	webdav
Site Manager Web Start	sitemanager
Studio	studio

4.2.1 Common Configurations

JMX

All CoreMedia web applications support JMX and register MBeans for management and monitoring purposes (see Section 4.9, “JMX Management” in *CoreMedia Operations Basics*). In order to reuse the remote connector server provided by WebSphere, make sure that the property `management.server.remote.url` is left empty.

To open a JMX connection to WebSphere from within a *CoreMedia* application (like the *Probedog* or the *CAE Developer Toolbox* in *Studio*), you have to use the `iiop` protocol, that is `service:jmx:iiop:///jndi/JMXConnector` and set the following JVM properties:

Table 4.3. JMX Configuration

key	`java.naming.provider.url`
value	`corbaloc:iiop:<YOUR_WEBSPHERE_HOST>/WsnAdminNameService`
key	`java.naming.factory.initial`
value	`com.sun.jndi.cosnaming.CNCtxFactory`

4.2.2 Application Specific Configuration

Content Management Server

Table 4.4. Content Management Server

`WEB-INF/properties/corem/contentserver.properties`	
key	`cap.server.http.port`
example value	`9080`
description	The HTTP port of the IBM WebSphere Application Server.
key	`cap.server.workflow.server.url`
example value	`http://localhost:9080/wfs/ior`
description	The URL of the IOR of the *Workflow Server*.
`WEB-INF/properties/corem/publisher.properties`	
key	`publisher.target.ior.url`
example value	`http://localhost:9080/mls/ior`

description	The URL of the login service IOR of the publication server.
`WEB-INF/config/contentserver/spring/search/search-solr.properties`	
key	`search.solr.urls`
example value	`http://localhost:9080/solr/studio`
description	The URL of the Solr search engine

Master Live Server

Table 4.5. Master Live Server

`WEB-INF/properties/corem/contentserver.properties`	
key	`cap.server.http.port`
example value	`9080`
description	The HTTP port of the IBM WebSphere Application Server.

Replication Live Server

Table 4.6. Replication Live Server

`WEB-INF/properties/corem/contentserver.properties`	
key	`cap.server.http.port`
example value	`9080`
description	The HTTP port of the IBM WebSphere Application Server.
`WEB-INF/properties/corem/replicator.properties`	
key	`replicator.publicationIorUrl`
example value	`http://localhost:9080/mls/ior`
description	The URL of the login service IOR of the publication server.

Workflow Server

Table 4.7. Workflow Server

`WEB-INF/properties/corem/capclient.properties`	
key	`cap.client.server.ior.url`
example value	`http://localhost:9080/cms/ior`

description	The URL of the login service IOR of the *Content Management Server*.

Content Feeder

Table 4.8. Content Feeder

`WEB-INF/application.properties`	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the *Content Management Server*.
key	`feeder.solr.url`
example value	`http://localhost:9080/solr/studio`
description	The URL of the Solr search engine for the Studio search.

User Changes

Table 4.9. User Changes

`WEB-INF/application.properties`	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the content management server.

Elastic Worker

Table 4.10. Elastic Worker

`WEB-INF/application.properties`	
This file needs to be created as the default values are resolved from classpath	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the content management server.
key	`elastic.solr.url`
example value	`http://localhost:9080/solr/`
description	The URL of the Solr Search Engine.

Solr Search Engine

Table 4.11. Solr search engine Master and Slave

WEB-INF/classes/logging.properties	
key	`org.apache.juli.FileHandler.directory`
example value	`/var/log/coremedia`
description	The logfile path

Solr Home Configuration

To install the Solr configuration, simply unzip the `modules/search/solr-config/target/solr-config.zip` file to an arbitrary location on your target host and make sure that you set the `solr.solr.home` JVM property accordingly in your WebSphere Application Server as described in Chapter 5, *Settings for IBM WebSphere Application Server* [36].

Alternatively, if you don't want to use JVM properties, which require a server restart, you may duplicate the `web.xml` file of the Solr web application and comment in the part where an `env-entry` is being used to set `solr.solr.home`.

The configuration for a master Solr only differs from the configuration for a slave Solr in the properties `enable.master` and `enable.slave` in the `solr-home/configsets/*/conf/solrconfig.xml` files.

The data directory can be set in the `solrconfig.xml` files too. Look for the `dataDir` XML element.

CAE Feeder Preview

Table 4.12. CAE Feeder Preview

WEB-INF/application.properties	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the content management server.
key	`feeder.solr.url`
example value	`http://localhost:9080/solr/preview`
description	The URL of the Solr search engine for the Preview search.

CAE Feeder Live

Table 4.13. CAE Feeder Live

`WEB-INF/application.properties`	
key	`repository.url`
example value	`http://localhost:9080/mls/ior`
description	The URL of the login service IOR of the *Master Live Server*.
key	`feeder.solr.url`
example value	`http://localhost:9080/solr/live`
description	The URL of the Solr search engine for the Preview search.

Preview CAE

Table 4.14. Preview

`WEB-INF/application.properties`	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the content management server.
key	`solr.search.url`
example value	`http://localhost:9080/solr/preview`
description	The URL of the Solr search engine for the Preview search.
key	`elastic.solr.url`
example value	`http://localhost:9080/solr/preview`
description	The URL of the Solr search engine for *Elastic Social* search.
key	`blueprint.sitemap.target.root`
example value	`${user.home}/cms/sitemap`
description	The root directory where the sitemap controller writes its files
key	`blueprint.host.helios`
example value	`localhost`
description	The host of the Preview CAE which is used when redirecting a preview request from Studio
key	`blueprint.host.studio.helios`

example value	`localhost`
description	The host of Studio
key	`blueprint.site.mapping.helios`
example value	`http://${blueprint.host.helios}:9080`
description	The site mapping if the web application needs to build absolute URLs
key	`repository.heapCacheSize`
example value	`314572800`
description	This property indicates the number of bytes used for the main memory cache of the *Unified API* embedded in the *Content Application Engine* . For 64-bit JVMs, the actual memory consumption may be up to twice the configured value. For 32-bit JVMs, the byte count is exact. When multiple CAEs run in a single application server, the caches are kept separate and the configured cache sizes add up.
key	`repository.blobCacheSize`
example value	`10737418240`
description	This property defines the number of bytes of the disk cache for blobs.
key	`repository.blobStreamingSizeThreshold`
example value	`-1`
description	The minimum size of streamed blobs in bytes. Blobs less than or equal to this size will be downloaded completely to disk before the first byte can be read. Larger blobs will be downloaded in the background.
key	`repository.blobStreamingThreads`
example value	`-1`
description	The number of threads reserved for streaming blob
key	`repository.maxCachedBlobSize`
example value	`-1`
description	The maximum size of blobs that are cached on the local disk. Larger blobs are downloaded from the *Content Server* on every request.

Delivery CAE

Table 4.15. Delivery

`WEB-INF/application.properties`

key	`repository.url`
example value	`http://localhost:9080/mls/ior`
description	The URL of the login service IOR of the *Master Live Server*.
key	`solr.search.url`
example value	`http://localhost:9080/solr/live`
description	The URL of the Solr search engine for the Live search.
key	`elastic.solr.url`
example value	`http://localhost:9080/solr/`
description	The URL of the Solr search engine for *Elastic Social* search.
key	`blueprint.sitemap.target.root`
example value	`${user.home}/cms/sitemap`
description	The root directory into which the sitemap controller writes its files
key	`blueprint.host.helios`
example value	`localhost`
description	The host on which the delivery CAE runs
key	`blueprint.site.mapping.helios`
example value	`http://${blueprint.host.helios}:9080`
description	The site mapping if the web application needs to build absolute urls
key	`repository.heapCacheSize`
example value	`314572800`
description	This property indicates the number of bytes used for the main memory cache of the *Unified API* embedded in the *Content Application Engine* . For 64-bit JVMs, the actual memory consumption may be up to twice the configured value. For 32-bit JVMs, the byte count is exact. When multiple CAEs run in a single application server, the caches are kept separate and the configured cache sizes add up.
key	`repository.blobCacheSize`
example value	`10737418240`
description	This property defines the number of bytes of the disk cache for blobs.
key	`repository.blobStreamingSizeThreshold`

example value	`-1`
description	The minimum size of streamed blobs in bytes. Blobs smaller than or equal to this size will be downloaded completely to disk before the first byte can be read. Larger blobs will be downloaded in the background.
key	`repository.blobStreamingThreads`
example value	`-1`
description	The number of threads reserved for streaming blob
key	`repository.maxCachedBlobSize`
example value	`-1`
description	The maximum size of blobs that are cached on the local disk. Larger blobs are downloaded from the *Content Server* on every request.

Studio

Table 4.16. Studio

`WEB-INF/application.properties`	
key	`repository.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the *Content Management Server*.
key	`studio.previewUrlPrefix`
example value	`http://localhost:9080/preview/servlet`
description	The URL of the Preview
key	`toolbox.jmx.url`
example value	`service:jmx:iiop:///jndi/JMXConnector`
description	The JMX connection address for the CAE Toolbox Plugin.
key	`es.cae.http.port`
example value	`9080`
description	The port of the CAE to which the user sends requests
key	`es.cae.http.host`
example value	`localhost`
description	The host of the CAE to which the user sends requests

key	`es.cae.protocol`
example value	`http`
description	The protocol used by the CAE for communication between user and CAE

key	`elastic.solr.url`
example value	`http://localhost:9080/solr/`
description	Apache Solr URL

key	`blueprint.host.helios`
example value	`localhost`
description	The host on which the preview CAE runs

key	`externalpreview.restUrl`
example value	`http://localhost:9080/preview/servlet/externalpreview`
description	The URL the REST resource of the external preview should push the data to. The controller addressed here must belong to a preview CAE.

key	`externalpreview.previewUrl`
example value	`http://localhost:9080/preview/externalpreview`
description	The URL where the login page of the external preview can be found.

key	`externalpreview.urlPrefix`
example value	`http://localhost:9080`
description	This prefix and the relative preview URL is send to the external preview URL to be displayed in the preview iframe.

key	`mongoDb.clientURI`
example value	`mongodb://localhost:27017/`
description	The standard MongoDB connection string URI is used to configure your MongoDB connection, for example, it allows you to configure read preferences and write concerns. The format of a client URI is documented under the following link: http://docs.mongodb.org/manual/reference/connection-string/.

key	`repository.heapCacheSize`
example value	`314572800`
description	This property indicates the number of bytes used for the main memory cache of the *Unified API* embedded in the *Content Application Engine* . For

	64-bit JVMs, the actual memory consumption may be up to twice the configured value. For 32-bit JVMs, the byte count is exact. When multiple CAEs run in a single application server, the caches are kept separate and the configured cache sizes add up.
key	`repository.blobCacheSize`
example value	`10737418240`
description	This property defines the number of bytes of the disk cache for blobs.
key	`repository.blobStreamingSizeThreshold`
example value	`-1`
description	The minimum size of streamed blobs in bytes. blobs less than or equal to this size will be downloaded completely to disk before the first byte can be read. Larger blobs will be downloaded in the background.
key	`repository.blobStreamingThreads`
example value	`-1`
description	The number of threads reserved for streaming blob
key	`repository.maxCachedBlobSize`
example value	`-1`
description	The maximum size of blobs that are cached on the local disk. Larger blobs are downloaded from the *Content Server* on every request.

WebDAV

Table 4.17. WebDAV

`WEB-INF/properties/corem/capclient.properties`	
key	`cap.client.server.ior.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the *Content Management Server*.

Site Manager

Table 4.18. Site Manager

`webstart/properties/corem/capclient.properties`	
key	`cap.client.server.ior.url`

example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the *Content Management Server*.
`webstart/properties/corem/editor.properties`	
key	`editor.configuration`
example value	`http://localhost:9080/editor-webstart/webstart/properties/corem/editor.xml`
description	The URL of the editor configuration. You cannot use localhost here, but instead use the fully qualified name of the host.
key	`editor.startup.configuration`
example value	`http://localhost:9080/editor-webstart/webstart/properties/corem/editor-startup.xml`
description	The URL of the editor startup configuration. You cannot use localhost here but instead use the fully qualified name of the host.
`webstart/properties/corem/workflowclient.properties`	
key	`workflow.client.server.ior.url`
example value	`http://localhost:9080/wfs/ior`
description	The URL of the Workflow Server
key	`cap.client.server.ior.url`
example value	`http://localhost:9080/wfs/ior`
description	The URL of the Workflow Server
`webstart/editor-webstart.jnlp`	
key	`javaws.coremedia.com.sun.CORBA.legacy.connection.ORBSocketFactoryClass`
default value	`com.sun.corba.se.impl.legacy.connection.DefaultSocketFactory`
description	Property to plug in an own socket factory class to an ORB
key	`javaws.coremedia.com.coremedia.corba.SSLClientSocketFactory.clearTextPort`
example value	`14300,14305`
description	"IIOP_CLEAR_TEXT" listening port of the server ORB, which is used in IOP profiles of object references exported by an ORB
key	`javaws.coremedia.com.coremedia.corba.SSLClientSocketFactory.sslPort`

example value	0
description	The SSL Port for the socket factory
key	`javaws.coremedia.com.coremedia.corba.SSLClientSocketFactory.keystore`
example value	
description	Path to the KeyStore
key	`javaws.coremedia.com.coremedia.corba.SSLClientSocketFactory.passphrase`
example value	
description	The password for the connection

Command Line Tools

The tools artifacts are located in the `modules/cmd-tools` folder and are named with the pattern `*-application`. Each of these modules create a `target/*-tools.zip` archive, that contains its tools. To deploy these artifacts, extract them and configure the properties below `properties/corem` to match your deployment.

Table 4.19. Tools

`properties/corem/capclient.properties`	
key	`cap.client.server.ior.url`
example value	`http://localhost:9080/cms/ior`
description	The URL of the login service IOR of the *Content Server*. Make sure you use the correct host and context path.
`properties/corem/sql.properties`	
key	`sql.store.driver`
example value	`org.postgresql.Driver`
description	The class of the database driver
key	`sql.store.url`
example value	`jdbc:postgresql://localhost:5432/coremedia`
description	The URL of the database connection
key	`sql.store.user`
example value	`cm7management`

description	The name of the database user

key	`sql.store.password`
example value	`cm7management`
description	The password of the database user

5. Settings for IBM WebSphere Application Server

In this chapter, you will find required settings for the deployment of the CoreMedia system into IBM WebSphere Application Server:

- Settings for the operating system of IBM WebSphere
- JVM properties
- Property for better performance
- The order of the class loaders
- ORB configuration properties
- Java Authentication and Authorization Service (JAAS) properties

OS Level Settings

For the operation system on which IBM WebSphere runs, you have to set the following properties at least to the specified values:

Table 5.1. Operating system settings

Key	Value
`process limit`	5000
`filelimit limit`	25000

JVM Settings

For a single WebSphere instance with all Blueprint applications deployed, you need at least the following memory settings:

Table 5.2. JVM Settings

Key	Value	Description
`initial-HeapSize`	4096	Initial memory for JVM
`maximum-HeapSize`	6144	Maximum memory for JVM

JVM System Properties

You have to set the following JVM system properties in IBM WebSphere as described below.

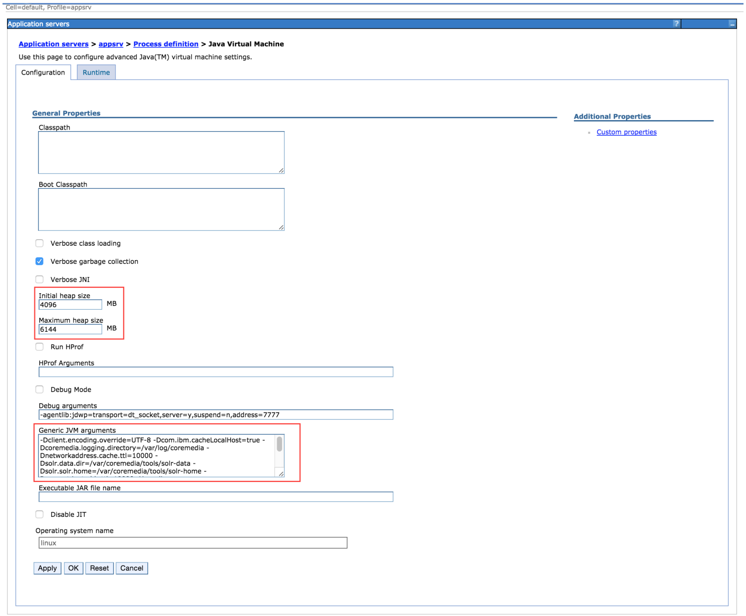
Table 5.3. JVM System Properties

key	`solr.solr.home`
example value	`/var/coremedia/solr-home`
description	Directory where Solr configuration and plugin settings are stored. Instead of setting this value here, which requires a restart of WebSphere, you can set it within the `web.xml` file of the *Solr* web application. A more detailed description can be found in Section "Solr Home Configuration" [26].
key	`client.encoding.override`
value	`UTF-8`
description	When deployed in a WebSphere server, Solr does not handle non-ASCII characters correctly (such as German umlauts ä, ö, ü). In order to fix this, add this property.
key	`coremedia.logging.directory`
example value	`/var/log/coremedia`
description	Directory where CoreMedia component logs are stored.
key	`com.coremedia.orb.jndiName`
value	`java:comp/ORB`
description	The communication between CoreMedia components is based on CORBA. It is recommended to inject the ORB provided by WebSphere into all deployed CoreMedia servers and UAPI clients. Alternatively to JVM system properties, you can add this property in the `application.properties` file, located in the `WEB-INF` folder of each web application. Make sure that if you set this property, that the host on this port is resolvable by all clients. section "Port Settings" [38] describes where you set this value.

To set the memory settings and JVM system properties, follow this click path:

Servers → **Server Types** → **WebSphere application servers** → **Server Name** → **Java and Process Management** → **Process definition** → **Java Virtual Machine**

Enter the JVM system properties into the *Generic JVM arguments* field as shown in Figure 5.1, “Define JVM Arguments in WebSphere” [38].



Application servers > appsrv > Process definition > **Java Virtual Machine**

Use this page to configure advanced Java(TM) virtual machine settings.

General Properties

Classpath

Boot Classpath

□ Verbose class loading

☑ Verbose garbage collection

□ Verbose JNI

Initial heap size: 4096 MB

Maximum heap size: 6144 MB

□ Run HProf

HProf Arguments

□ Debug Mode

Debug arguments: -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777

Generic JVM arguments:

```
-Dclient.encoding.override=UTF-8 -Dcom.ibm.cacheLocalHost=true -
Dcoremedia.logging.directory=/var/log/coremedia -
Dnetworkaddress.cache.ttl=10000 -
Dsolr.data.dir=/var/coremedia/tools/solr-data -
Dsolr.solr.home=/var/coremedia/tools/solr-home -
```

Executable JAR file name

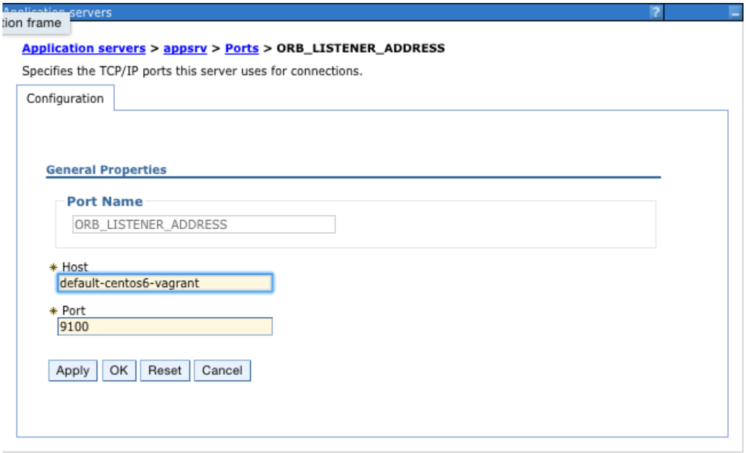
□ Disable JIT

Operating system name: linux

Figure 5.1. Define JVM Arguments in WebSphere

Port Settings

If you inject the IBM WebSphere ORB into all servers and UAPI clients, you need to make sure that the host name set for this port can be resolved by the client. You will find the ORB ports host setting at the following click path: **Servers** → **Server Types** → **WebSphere application servers** → **Server Name** → **Ports** → **ORB_LISTENER_ADRESS**



Application servers > appsrv > Ports > ORB_LISTENER_ADDRESS

Specifies the TCP/IP ports this server uses for connections.

Configuration

General Properties

Port Name
ORB_LISTENER_ADDRESS

* Host
default-centos6-vagrant

* Port
9100

Figure 5.2. Set full qualified name for ORB port

Security Settings

CoreMedia command line clients, as installed by the standard deployment, are not able to connect to a *Content Server* running in WebSphere Application Server using an SSL encrypted CORBA connection. Therefore, in order to use these tools, the WAS must also accept unencrypted communication. Use the following steps to configure the server:

1. In WebSphere go to the following window:

   Servers → **Server Types** → **WebSphere application servers** → **Server Name** → **Security** → **Security Domain** → **RMI/IIOP security** → **CSIv2 inbound communication** → **Transport**

2. Change the entry from **SSL-required** to **SSL-supported**.

Now, the clients communicate with the *Content Server* in the WebSphere Application Server through an unsecured CORBA connection. However, all clients that support SSL encoding use a secure connection.

For additional security, remote access to the clear-text CORBA port can be restricted using operating system and firewall means.

Improve Performance

For better performance of the CoreMedia web applications deployed to the WebSphere Application Server, it is highly recommended to set the following property in the same place as the properties described in the previous section.

→ **Set local hostname caching to true:**

```
com.ibm.cacheLocalHost=true
```

Class loader Order

CoreMedia components can be successfully deployed in a WebSphere environment using the class loader configuration `PARENT_LAST`, set for each web application module.

In order to enable this class loader order, select in the WebSphere Administration Console:

Applications → WebSphere enterprise applications → Application Name → Manage Modules → Classes loaded with local class loader first (parent last)

Java Authentication and Authorization Service (JAAS)

Creating a custom JAAS login module

CoreMedia content servers use a custom JAAS Login module, named JaasCap with the settings, described in Table 5.4, “ Java Authentication and Authorization Service (JAAS) ” [40]. In Create JAAS module [41] is described how you have to create the module.

Table 5.4. Java Authentication and Authorization Service (JAAS)

Key	Value
`loginType`	application
`loginModules`	hox.corem.server.CapLoginModule
`authStrategies`	SUFFICIENT

The login module requires the following predicates as custom properties:

Table 5.5. Custom JAAS LoginModule Properties

Key	Value
`predicate.1.class`	hox.corem.login.NameLoginPredicate
`predicate.1.args`	negative=true,editor.regex=(serverdump|publisher|auto-actor|watchdog|workflow|webserver|importer|feeder), filesystem.regex=(serverdump|publisher|auto-actor|watchdog|workflow|webserver|importer|feeder)
`predicate.2.class`	hox.corem.login.NameLoginPredicate
`predicate.2.args`	webserver.regex=webserver,publisher.regex=publisher,replicator.regex=replicator,workflow.regex=workflow,feeder.regex=feeder
`predicate.3.class`	hox.corem.login.NameLoginPredicate

Key	Value
`predicate.3.args`	editor.regex=.*,debug.regex=.*,filesystem.regex=.*,importer.regex=.*,system.regex=.*

Create the JAAS login module with the settings given above as follows:

Create JAAS module

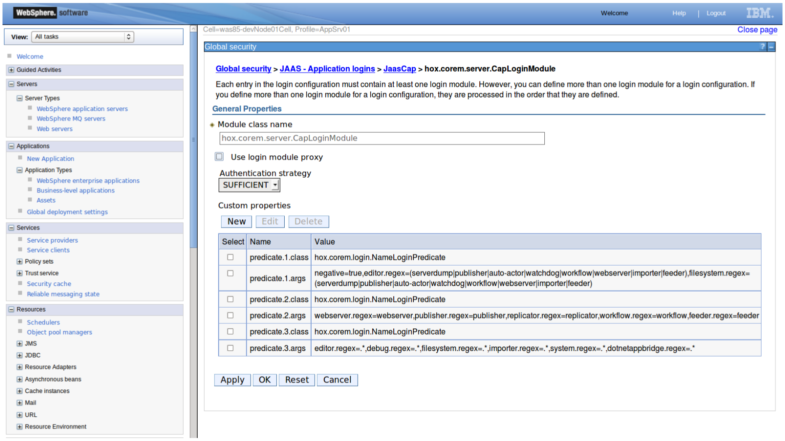
1. Create a custom Login Module named `JaasCap` with class name `hox.corem.server.CapLoginModule` in the WebSphere Administration Console under:

   Security → Global Security → Java Authentication and Authorization Service → Application logins

2. Set its authentication strategy to `SUFFICIENT`.

3. The `JAAS` predicates are located in the `jaas.conf` file, under `WEB-INF/properties/corem` of a Content Server's installation folder.

   Add each predicate as a custom property to the newly configured `hox.corem.server.CapLoginModule` in WebSphere (see Figure 5.3, “Custom JAAS LoginModule” [41]).



Cell=was85-devNode01Cell, Profile=AppSrv01

Global security > JAAS - Application logins > JaasCap > hox.corem.server.CapLoginModule

Each entry in the login configuration must contain at least one login module. However, you can define more than one login module for a login configuration. If you define more than one login module for a login configuration, they are processed in the order that they are defined.

General Properties

* Module class name: hox.corem.server.CapLoginModule

□ Use login module proxy

Authentication strategy: SUFFICIENT

Custom properties

Select	Name	Value
□	predicate.1.class	hox.corem.login.NameLoginPredicate
□	predicate.1.args	negative=true,editor.regex=(serverdump|publisher|auto-actor|watchdog|workflow|webserver|importer|feeder),filesystem.regex=(serverdump|publisher|auto-actor|watchdog|workflow|webserver|importer|feeder)
□	predicate.2.class	hox.corem.login.NameLoginPredicate
□	predicate.2.args	webserver.regex=webserver,publisher.regex=publisher,replicator.regex=replicator,workflow.regex=workflow,feeder.regex=feeder
□	predicate.3.class	hox.corem.login.NameLoginPredicate
□	predicate.3.args	editor.regex=.*,debug.regex=.*,filesystem.regex=.*,importer.regex=.*,system.regex=.*,dotnetappbridge.regex=.*

Figure 5.3. Custom JAAS LoginModule

Web Container Settings

In order to run CAE web applications in IBM WebSphere you have to set two properties for the web container in WebSphere. For each property create a new custom property in the WebSphere Administration Console under: **Application servers → Server Name → Web container → Custom properties**

Table 5.6. Web Container Settings

key	`com.ibm.wsspi.jsp.evalQuotedAndEscapedExpression`
example value	`true`
description	There is a known issue in WebSphere's environment concerning the evaluation of Taglib functions within single quotes. Set this property to true in order to enable this functionality.
key	`com.ibm.ws.webcontainer.extractHostHeaderPort`
example value	`true`
description	Set the trusthostheaderport and the com.ibm.ws.webcontainer.extractHostHeaderPort custom property to true to return the port number from the request host header first. http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/rweb_custom_props.html?cp=SSEQTP_8.5.5%2F1-17-5-994
key	`trusthostheaderport`
example value	`true`
description	Set the trusthostheaderport and the com.ibm.ws.webcontainer.extractHostHeaderPort custom property to true to return the port number from the request host header first. http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/rweb_custom_props.html?cp=SSEQTP_8.5.5%2F1-17-5-994
key	`httpsIndicatorHeader`
example value	`X-Forwarded-HTTPS`
description	The SSL offloader must be configured to add a special header indicating that the original request was over HTTPS. On the proxy / loadbalancer, make sure to inject this header as request header. See http://www-01.ibm.com/support/docview.wss?uid=swg21221253 for a detailed description. Add for example this to your virtualhost configuration for Apache: `RequestHeader set X-Forwarded-HTTPS "true"`

6. Deploying CoreMedia Collaborative Components without MongoDB

CoreMedia components provide collaborative features such as projects and todos, notifications or Studio workflows for publication and translation. The CoreMedia components which provide these features are Studio, Workflow Server and User Changes web applications. By default, these components use MongoDB as the persistence layer for the collaborative features. However, it is possible to use an in-memory setup instead. We refer to the section `In-Memory Replacement for MongoDB-Based Services` of the [*CoreMedia DXP 8* Manual].

7. Known Limitations

Be aware of the following limitations when deploying and operating CoreMedia components in IBM WebSphere Application Server.

- The *Watchdog* web application is currently not supported for usage in IBM WebSphere.
- Standalone Elastic Worker web application

  The Elastic Worker web application is currently not supported for WebSphere Application Server network deployment. To use the Elastic Worker features, copy all JAR files except `xpp-XXX.jar` from the `WEB-INF/lib` folder of the Elastic Worker to the preview Blueprint web application library folder. Set the property `taskQueues.workerNode` to "true" in Blueprint's `application.properties` file to make the preview CAE a worker node.