

CoreMedia Digital Experience Platform 8
//Version 7.5.45-10

COREMEDIA 

CoreMedia Content Server Manual

COREMEDIA



CoreMedia Content Server Manual

Copyright CoreMedia AG © 2015

CoreMedia AG

Ludwig-Erhard-Straße 18

20459 Hamburg

International

All rights reserved. No part of this manual or the corresponding program may be reproduced or copied in any form (print, photocopy or other process) without the written permission of CoreMedia AG.

Germany

Alle Rechte vorbehalten. CoreMedia und weitere im Text erwähnte CoreMedia Produkte sowie die entsprechenden Logos sind Marken oder eingetragene Marken der CoreMedia AG in Deutschland. Alle anderen Namen von Produkten sind Marken der jeweiligen Firmen.

Das Handbuch bzw. Teile hiervon sowie die dazugehörigen Programme dürfen in keiner Weise (Druck, Fotokopie oder sonstige Verfahren) ohne schriftliche Genehmigung der CoreMedia AG reproduziert oder vervielfältigt werden. Unberührt hiervon bleiben die gesetzlich erlaubten Nutzungsarten nach dem UrhG.

Licenses and Trademarks

All trademarks acknowledged.
07.Mar 2017

1. Introduction	1
1.1. Audience	2
1.2. Typographic Conventions	3
1.3. CoreMedia Services	5
1.3.1. Registration	5
1.3.2. CoreMedia Releases	5
1.3.3. Documentation	6
1.3.4. CoreMedia Training	8
1.3.5. CoreMedia Support	9
1.4. Change Chapter	12
2. Overview	13
2.1. The Content Server	14
2.2. Replication Live Servers	15
2.2.1. State Diagram	15
2.2.2. Extent of Replication	17
2.2.3. Inconsistencies during Replication	17
2.2.4. Fault Tolerance of the Content Delivery Environ- ment	18
2.3. Multi-Site Publishing	23
2.4. Server Run Levels	25
2.5. Changelog	27
3. Configuration and Operation	28
3.1. Structure of Content Server Installation	29
3.2. Deploying the Content Server	30
3.3. Configuring the Database	31
3.3.1. Specifying Tablespaces for the Content Serv- er	31
3.3.2. Oracle Database	34
3.3.3. IBM DB2 Database	36
3.3.4. Microsoft SQL Server	36
3.3.5. PostgreSQL Database	36
3.3.6. MySQL Database	38
3.4. Configuring Blob Storage	39
3.5. Exclusive Locks	45
3.6. Configuring CORBA	46
3.7. Extending the Content Server	47
3.8. Starting the Server	48
3.9. Recovery of Content Server Databases	50
3.9.1. Backup Strategy	50
3.9.2. Recovery of a Content Management Server Database	51

3.9.3. Recovery of a Master Live Server Database	51
3.10. Administrating Replication Live Servers	54
3.10.1. Installing the First Replication Live Server	54
3.10.2. Installing further Replication Live Servers	55
3.10.3. Replication Live Servers Backups	55
3.10.4. Restoring from Replication Live Server Backup	56
3.10.5. Restoring from Master Live Server Backup	56
3.10.6. Removing a Replication Live Server	60
3.10.7. Analyzing the Replicator State	60
3.11. Administrating Multi-Site Publishing	65
3.11.1. Enabling Multi-Site Publishing	65
3.11.2. Configuring Multi-Site Publishing	65
3.11.3. Adding Publication Targets	66
3.11.4. Migrate to Multi-Site Management Extension	67
3.12. Truncate the ChangeLog	75
3.13. LDAP Integration	77
3.13.1. User Authentication	77
3.13.2. Configuration of UserProviders	84
3.13.3. LdapUserProvider	85
3.13.4. ActiveDirectoryUserProvider	86
3.13.5. Connecting LDAP Over SSL	89
3.14. Server Utility Programs	90
3.14.1. Information	91
3.14.2. Operation	116
3.14.3. Repository	142
3.15. JMX Management	159
3.16. User Administration	160
3.16.1. Predefined Users and Groups	160
3.16.2. User Rights Management	163
3.16.3. Administrator Groups	177
3.16.4. Content Server Groups and Users	178
3.16.5. Live Server Groups and Users	178
3.16.6. Managing Users	180
3.17. WebDAV Support	270
3.17.1. Concepts	191
3.17.2. Configuration and Operation	194
3.18. Troubleshooting	198
4. Developing a Content Type Model	200

4.1. Properties	202
4.2. Creating Content Type Definitions	208
4.2.1. Structure of Content Type Definitions	209
4.2.2. Inheriting Content Types	213
4.2.3. Attaching Properties to Existing Content Types	214
4.3. Schema Migration	216
4.3.1. The Database Schema	216
4.3.2. Adding Content Types	218
4.3.3. Renaming Content Types	219
4.3.4. Deleting Content Types	220
4.3.5. Adding Properties	221
4.3.6. Renaming Properties	221
4.3.7. Deleting Properties	223
5. Appendix	226
5.1. Configuration in contentserver.properties	227
5.2. Configuration in publisher.properties	234
5.3. Configuration in sql.properties	240
5.3.1. Overview of all Properties	240
5.4. Configuration in replicator.properties	247
5.5. Configuration in capclient.properties	250
5.6. Reference of webdav.properties	251
5.7. Managed Properties	259
Glossary	265
Index	272

List of Figures

2.1. Replication Live Server state diagram	16
3.1. Result of a command in Groovy Shell	129
3.2. CM SQL user interface	139
3.3. User administration window	180
3.4. Administrator Group	181
3.5. Creating a new group	182
3.6. Assign rules for group	183
3.7. Creating a user profile	183
3.8. Rights structure of a resource	185
3.9. Rules of the user for a resource	185
3.10. The File menu (administrator view)	187
3.11. Example: WebDAV file	193
4.1. Schema of content type definitions	208
4.2. Extending content types	215

List of Tables

1.1. Typographic conventions	3
1.2. Pictographs	3
1.3. CoreMedia manuals	6
1.4. Log files check list	10
1.5. Changes	12
2.1. Master Live Server failure	19
2.2. Master Live Server deadlock	20
2.3. Database failure	20
2.4. Slave Live Server failure	21
2.5. Database failure	21
2.6. CAE failure	21
2.7. CAE deadlock	22
3.1. Attributes of the bean element	42
3.2. Condition classes which could be used in the <bean> element.	42
3.3. Attributes of the property element	44
3.4. Properties used to configure Multi-Site Management	65
3.5. Options of CapLoginModule	79
3.6. Options of the LdapLoginModule	79
3.7. CoreMedia services	80
3.8. NameLoginPredicate options	81
3.9. AttributeLoginPredicate options	82
3.10. JndiNameLoginPredicate options	83
3.11. Common options of server utilities	90
3.12. Options of dump	92
3.13. Options of the events utility	97
3.14. Options of license	98
3.15. The parameters of processorusage	100
3.16. Options of the publications utility	100
3.17. Options of the rules utility	102
3.18. Options of the session utility	103
3.19. Parameters of validate-multisite	104
3.20. Issues of validate-multisite	105
3.21. Options of the changepassword tool	118
3.22. Options of the clean recycle bin	120
3.23. Switches of the version collector	122
3.24. Options of dbindex	127
3.25. Options of dumpusers	128

3.26. Options of jmxdump	130
3.27. Parameters of restoreusers	132
3.28. Options of runlevel	132
3.29. Schemaaccess actions	134
3.30. Parameters of the serverimport utility	135
3.31. Parameters of the serverexport utility	136
3.32. Options of tracesession	140
3.33. Parameters of the usedlicenses utility	142
3.34. Options of approve	144
3.35. Options of the bulkpublish tool	145
3.36. Options of destroy	146
3.37. Options of publish	148
3.38. Parameters of the publishall utility	149
3.39. Parameters of the republish utility	151
3.40. Parameters of the query utility	153
3.41. Identifiers and literals	156
3.42. Parameters of the queryapprove utility	157
3.43. Parameters of the querypublish utility	158
3.44. Parameters of the search utility	158
3.45. Standard groups	160
3.46. Users and their groups	161
3.47. Users and their groups in Content Management Server only	161
3.48. Default mapping of user rights	161
3.49. Workflow role groups	163
3.50. User rights	164
3.51. Example rule table	165
3.52. Rule to read a content item	166
3.53. Rule to create a content item	166
3.54. Rule for content item operations	167
3.55. Rules to move a content item	167
3.56. Rule to mark or (un)mark a content item for deletion	167
3.57. Rules to delete a content item	168
3.58. Rule to (dis)approve a content item	168
3.59. Rule to publish a content item	168
3.60. Rule to check in content items of other users	169
3.61. Rule to read folder properties	169
3.62. Rule to place approve or disapprove a folder	170
3.63. Rule to publish a folder	170
3.64. Rule to create subfolders	170
3.65. Rule to operate on subfolders	171

3.66. Rules to move a folder	171
3.67. Rule to supervise a content item	172
3.68. Rule to supervise content items in a folder	172
3.69. Rule to supervise a folder	172
3.70. Example rules for rights computation	173
3.71. Example for conflicting rules	174
3.72. Example rules to compute effective rights	175
3.73. Example rules with implicit navigate through right	176
3.74. Example rules with resolved navigate through right	176
3.75. Example rules with implicit READ right	176
3.76. Example rules with explicit READ right	176
3.77. Example rules with READ right withdrawn	177
3.78. Example groups	186
4.1. Attributes of a StringProperty field	203
4.2. Attributes of an XmlProperty field	203
4.3. Attributes of LinkListProperty	204
4.4. Attributes of XmlGrammar element	211
4.5. Attributes of the XMLSchema element	212
4.6. Attributes of the ImportGrammar element	213
4.7. System attributes of the content type table	217
4.8. How to delete properties of different type	224
5.1. contentserver.properties	227
5.2. contentserver.properties	232
5.3. contentserver.properties	233
5.4. publisher.properties	234
5.5. Properties for configuration of database connection	240
5.6. Properties for configuration of database schema	241
5.7. Properties for the configuration of the XML property conversion	243
5.8. Properties for configuration of blob and SgmlText collector	244
5.9. Properties to configure the SQL connection Pool	244
5.10. replicator.properties	247
5.11. capclient.properties	250
5.12. HTTP authentication	251
5.13. Properties for exported content types	252
5.14. Properties for the session timeout	255
5.15. The properties for the information file	256
5.16. Properties for character encoding	257
5.17. Properties for further customization	257
5.18. JMX manageable attributes of the Content Server	259

5.19. JMX manageable operations of the Content Server	260
5.20. JMX manageable attributes of the Publisher	261
5.21. JMX manageable attributes for Publication Targets	262
5.22. JMX manageable operations of the Publisher	263
5.23. JMX manageable attributes of the Replicator	263

List of Examples

3.1. Structure of the Content Server web application	29
3.2. Oracle: create DB User	35
3.3. Add privileges for Oracle 12c user	35
3.4. Oracle: delete DB User	35
3.5. Example configuration	41
3.6. The jaas.conf file	78
3.7. JAAS syntax	78
3.8. jaas.conf example	84
3.9. Example for Active Directory Server	84
3.10. Configuration of two providers	84
3.11. Property file	85
3.12. Property file	85
3.13. Properties	86
3.14. Usage of the events utility	96
3.15. Output of events	97
3.16. Execution of CM IOR with IOR of the CM Server	98
3.17. Result of repositorystatistics	101
3.18. Usage of validate-multisite	104
3.19.	118
3.20. Example of a customized predicate	125
3.21.	127
3.22. Usage of the restoreusers tool	131
3.23. Usage of schemaaccess	133
3.24. CM sql command line operation	139
3.25. Usage of bulkpublish	145
3.26. Usage of publishall	149
3.27.	151
3.28. Query usage	152
3.29. Result of a query	154
3.30. Transformation of a legacy query	154
3.31. EBNF definition of the query language	154
3.32. Usage of search utility	158
3.33. Groups in workflowserver.properties	163
3.34. A sample login bouncer configuration	189
4.1. The base-doctypes.xml file	208
4.2. The specific-doctypes.xml file	209
4.3. Example of a content type definition	209
4.4. Using XML Schemas	212

4.5. Example for content type inheritance 214

1. Introduction

This manual describes the concepts and configuration of the *Content Servers*. That is the *Content Management Server*, the *Master Live Server* and the *Replication Live Server*.

- [Chapter 2, Overview \[13\]](#) describes the concepts and components of the *Content Servers*.
- [Chapter 3, Configuration and Operation \[28\]](#) describes how you configure and operate the *Content Servers*.
- [Chapter 4, Developing a Content Type Model \[200\]](#) describes how you develop a document type model for the *Content Servers*.
- [Chapter 5, Appendix \[226\]](#) lists the configuration files of the *Content Servers*.

Search service is configured and running in the *Content Server* but is described in the Search Manual.



1.1 Audience

This manual is intended for everyone who wants to get an overview over *Content Servers*. It is specifically intended for operators that want to operate *Content Servers*.

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:

Element	Typographic format	Example
Source code	Courier new	<code>cm systeminfo start</code>
Command line entries		
Parameter and values		
Class and method names		
Packages and modules		
Menu names and entries	Bold, linked with	Open the menu entry Format Normal
Field names	Italic	Enter in the field <i>Heading</i>
CoreMedia Components		The <i>CoreMedia Component</i>
Applications		Use <i>Chef</i>
Entries	In quotation marks	Enter "On"
(Simultaneously) pressed keys	Bracketed in "<>", linked with "+"	Press the keys <Ctrl>+<A>
Emphasis	Italic	It is <i>not</i> saved
Buttons	Bold, with square brackets	Click on the [OK] button
Code lines in code examples which continue in the next line	\	<code>cm systeminfo \ -u user</code>
Mention of other manuals	Square Brackets	See the [Studio Developer Manual] for more information.

Table 1.1. Typographic conventions

In addition, these symbols can mark single paragraphs:




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

Table 1.2. Pictographs

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:

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 <i>CoreMedia Studio</i> for editorial and administrative work. It also describes the usage of the <i>Adaptive Personalization</i> and <i>Elastic Social</i> GUI that are integrated into <i>Studio</i> .

Table 1.3. CoreMedia manuals

Manual	Audience	Content
LiveContext for IBM WebSphere Manual	Developers, architects, administrators	<p>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 <i>Studio</i> extensions, folder and user rights concept and many more details. It also describes administrative tasks for the features.</p> <p>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.</p>
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 <i>Adaptive Personalization</i> , the CoreMedia module for personalized websites. You will learn how to configure the GUI used in <i>CoreMedia Studio</i> , 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 <i>Content Application Engine (CAE)</i> . 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 <i>Content Server</i> . 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 <i>Elastic Social</i> 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 <i>Importer</i> 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 <i>CoreMedia Search Engine</i> and the two feeder applications: the <i>Content Feeder</i> and the <i>CAE Feeder</i> .
Site Manager Developer Manual	Developers, architects, administrators	This manual describes the configuration and customization of <i>Site Manager</i> , the Java based stand-alone application for administrative tasks. You will learn how to configure the <i>Site Manager</i> with property files and XML files and how to develop your own extensions using the <i>Site Manager API</i> .
Studio Developer Manual	Developers, architects	This manual describes the concepts and extension of <i>CoreMedia Studio</i> . You will learn about the underlying concepts, how to use the development environment and how to customize <i>Studio</i> to your needs.
Unified API Developer Manual	Developers, architects	This manual describes the concepts and usage of the <i>CoreMedia Unified API</i> , 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 <i>Workflow Server</i> . 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

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:

Support request

- 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:

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

Support checklist

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.

Log files

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.

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 pre-view	coremedia.log (Content Management Server) coremedia.log (Master Live Server)

Table 1.4. Log files check list

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) *.jpic 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.

Section	Version	Description
Section "Groovy Shell" [129]	7.5.43	Added description of <i>groovysh</i> utility.
Section "JMXDump" [130]	7.5.41	Added description of the <i>jmx-dump</i> utility
Section 3.3.6, "MySQL Database" [38]	7.5.28	Changed version of recommended MySQL driver.

Table 1.5. Changes

2. Overview

The *CoreMedia Content Server* is the central component in the CoreMedia system. Among other things, it manages the content repository and the user authorization. It comes in three flavors:

- The Content Management Server
- The Master Live Server(s)
- The Replication Live Server(s)

This chapter gives a description of the concepts of the *CoreMedia Content Server*.

2.1 The Content Server

The *CoreMedia Content Server* is the central component in the CoreMedia system. Among other things, it manages the content repository and the user authorization. Like all CoreMedia components it is based on Java technology. At least two *CoreMedia Content Servers* are required to run an online editorial system. The *CoreMedia Content Management Server* is the production system used to create and administrate content.

In the *Content Delivery Environment* the *Master Live Server* receives the approved content from the *Content Management Server* and makes it available to the *CAE* which generate websites or documents in other formats like PDF. If you are using the *CoreMedia Multi-Site Management Extension*, content from different top-level folders can be published to different *Master Live Servers*.

Content Delivery Environment

For highest stability and scalable performance, the *Content Delivery Environment* can be expanded by so called *Replication Live Servers* (see [Section 2.2, "Replication Live Servers" \[15\]](#)).

The *CoreMedia Content Servers* and the other CoreMedia components communicate using CORBA. To build up a CORBA connection with the server, a client first has to send an HTTP request to the server to get the IOR, which contains the necessary connection parameters. To be able to connect, each host involved in the CORBA connection must be able to resolve the name of the other host through DNS. The *Content Server* caches requested resources in memory, therefore reducing database queries and improving the performance.

CORBA communication

The following property files hold the server configuration:

- `contentserver.properties`
- `sql.properties`
- `publisher.properties` (*Content Management Server* only)
- `mime.properties`
- `replicator.properties` (*Replication Live Server* only)

Note: The property files may contain additional property entries which are not described in this Manual. As a rule, these properties are for special, system-relevant settings which should not be adjusted by the customer.

2.2 Replication Live Servers

The *Replication Live Server* is a complete server installation with its own database instance, like a *Master Live Server* installation. The *Replication Live Server* differs from a *Master Live Server* in the following points:

- A *Replication Live Server* needs a different license file, that defines the server type (live).
- For a *Replication Live Server* you must configure the `replicator.properties` file (see [Section 5.4, “Configuration in replicator.properties” \[247\]](#)).
- The content of a *Replication Live Server* is updated differently. The *Replication Live Server* is a replicated image of the *Master Live Server*. It receives changes from the *Master Live Server* and updates its database accordingly. In particular, it can track changes after an offline phase. The process responsible for content update is called the *replication process* or *replicator*.

The aim of a *Replication Live Server* is to remove load from the *Master Live Server* and enable a scalable delivery architecture. So, you can install any number of *Replication Live Servers*, each with its own database instance. The *Content Application Engines (CAE)* do not access the *Master Live Server* anymore but the RLS. Multiple *CAE/web server* pairs can be attached to one RLS.

Do not modify the content repository of the *Replication Live Server* directly, either through one of the command line tools or through *Site Manager* or *Studio*. All modification should be done by the automatic replication process. While the user `admin` is granted write rights, such rights should be exercised only when advised accordingly by CoreMedia support in exceptional circumstances.



2.2.1 State Diagram

The following state diagram holds not for the initial replication! The initial replication is not fault-tolerant against connection losses but has to run without interruptions in order to succeed.



The following diagram shows the different states of the *Replication Live Server*:

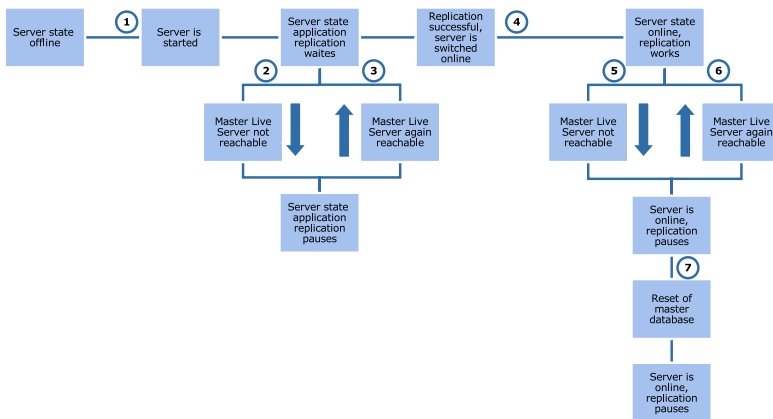


Figure 2.1. Replication Live Server state diagram

In the figure the transitions denoted with numbers in circles have the following meaning:

1. The server starts up and remains in administration mode. At this time, no *Content Application Engine (CAE)* can connect. The server connects with the *Master Live Server* and asks for changes. If there are changes, replication begins.
2. The *Master Live Server* is not reachable. It has either broken down, has been stopped, is working defectively, or communication is disrupted. In this case, replication pauses. Every 30 seconds, the *Replication Live Server* attempts to connect to the *Master Live Server*.
3. The *Replication Live Server* was able to reconnect to the *Master Live Server*. Replication starts again as described in 1.
4. Replication is finished. The *Replication Live Server* is switched to online mode. If the *Replication Live Server* is started in transition 1 for the first time, it does not switch to online mode automatically. Instead, you have to switch to online mode manually with `cm runlevel -u <user> -p <password> -r online -g 0`. Later the server switch is done automatically. Once the *Replication Live Server* is switched online, it continues to monitor the *Master Live Server* for changes and, if necessary, starts the replication.
5. As in 2., the connection to the *Master Live Server* is lost. Replication pauses, but the *Replication Live Server* remains online. Every 30 seconds, the *Replication Live Server* attempts to connect to the *Master Live Server*.
6. The *Replication Live Server* reconnects to the *Master Live Server* and replication starts again, as described in 1.
7. The database of *Master Live Server* has been reset. Replication stops and the *Replication Live Server* remains online. In order to synchronize with the *Master Live Server*, the database of *Replication Live Server* must also be reset.

2.2.2 Extent of Replication

The following information is replicated from the *Master Live Server*:

- created folders
- created content items
- created content versions
- destruction of content versions
- destruction of content items
- renaming/moving of content items and folders
- creation dates of folders, documents and versions: These dates are copied unchanged from the *Master Live Server*, indicating the time of the most recent publication of these objects. Note that some server tools, such as the `cm publishall` tool, may reset the publication date, so business logic should not rely too much on this metadata.

Certain information is not replicated:

- document editing state: By definition, there are only content items with state "published" on the *Master Live Server*. Nevertheless, the flags for approved and published still exist on the *Master Live Server*, but get lost during replication. This also applies to the associated date and user properties like approval date or publisher.
- content types: Any changes in the content type definitions must be carried out by hand on all server instances.
- user and group data: Any changes must be carried out by hand on all server instances.
- creation date in the management environment: As the creation date in the delivery environment indicates the time of the publication, the original creation date becomes inaccessible.

2.2.3 Inconsistencies during Replication

When a new content item is published on the *Master Live Server*, the replication process receives a change event and starts to read the content data from the *Master Live Server*. In rare cases, situations with potentially inconsistent data can arise:

- In the time period between the change event and the content data transmission the relevant content items is destroyed. To resolve this inconsistency the replication process waits until the destruction event for the corresponding content item has arrived.

- The content item does exist but in the time period between the change event and the content data transmission the relevant version was destroyed. Since there are only two content versions on a *Live Server*, this can only happen during a short period during publication or if a replicator which has not been running for a long period catches up. To fix this inconsistency, the replication process waits for the publication event of the new version and the deletion of the old version.

The replication process only resolves inconsistencies when the *Replication Live Server* is in "online" mode, since temporary inconsistencies must be avoided here.

2.2.4 Fault Tolerance of the Content Delivery Environment

In addition to scalability, the replication of content on more than one *Replication Live Servers* ensures system stability. A *Live Server*, that fails to operate, can be replaced by a running component. The following scenarios describe various types of system failures in an example Live system.

Example Live System Setup

This is the example setup used for the error scenarios that follow:

- CoreMedia Content Management Server (with CoreMedia Studio and CoreMedia CAE for preview)
- CoreMedia Master Live Server (Server)
- CoreMedia Replication Live Server 1 (Server)
- CoreMedia Replication Live Server 2 (Server)
- CoreMedia CAE 1-1
- CoreMedia CAE 1-2
- CoreMedia CAE 2-1
- CoreMedia CAE 2-2
- Database Instance of Content Management Server
- Database Instance of Master Live Server
- Database Instance of Replication Live Server 1
- Database Instance of Replication Live Server 2
- All CoreMedia components in this setup are monitored with their own *Watchdog*, running on the same machine as the components being monitored.

Alternatively, the production system and the *Master Live Server* can be monitored with the High Availability Software of the operating system.

You can install the production system with all its components on a single machine. As this chapter deals with failures on the Live system, it is not affected and it is only mentioned for the sake of completeness. The *Master Live Server* and the *Replication Live Servers* 1 and 2 are each installed on their own machines. If one machine breaks down, only the installation on that machine is affected. Each of the two *Replication Live Servers* processes the requests of two *CAEs*, which ideally should each be installed on their own machines. The *CAE* answers requests from a load balancing system, the latter is not covered here.

The database instances ought to have their own separate installations, distribution over several computers is even more secure. Security against breakdown of the database must be guaranteed by mechanisms of the database system or operating system.

Malfunctions of the Master Live Server

This scenario describes a *Master Live Server* malfunction. This error influences two processes:

- publication: Publications are canceled as defective, blocked or cannot be carried out because the *Master Live Server* is not reachable.
- replication: Replication is disabled. The *Replication Live Servers* stay online and the connected *CAEs* continue to operate with the already replicated content base.

Possible errors are:

Master Live Server Failure	
Error behavior	<p>The <i>Replication Live Servers</i> connected to the <i>Master Live Server</i> discontinues replication and attempts to log on again to the <i>Master Live Server</i> periodically in order to continue replication. Log attempts are written to the server log.</p> <p>Publications are no longer possible and the publication which was running at the time of disruption fails. You will see an error message if you try to start a publication or publication preview with <i>CoreMedia Studio</i>.</p>
Error correction	<p>In an environment without cluster, the watchdog of the <i>Master Live Server</i> detects the failure and restarts the server. In the cluster environment, the cluster software controls the watchdog (probedog). After server restart, the <i>Replication Live Server</i> log onto the <i>Master Live Server</i> and publications are possible again.</p>

Table 2.1. Master Live Server failure

Master Live Server Deadlock	
Error behavior	The clients hang up or receive error messages until the watchdog restarts the server.
Error correction	The watchdog detects the error and restarts the server. Restarting the server implies a brief server failure. Therefore, the further behavior of the clients corresponds to the behavior described under <i>Live Server Failure</i> .

Table 2.2. Master Live Server deadlock

Master Live Server Database Failure	
Error behavior	Transactions which are active at the time of failure or which first notice the failure are terminated with an error. The error is passed to the server and clients. Further publications lead to errors. The <i>Replication Live Servers</i> detect the malfunction and attempt to reconnect. You can see the login attempts in the server log. Transactions started after the server has detected the database failure are blocked until a new database connection is created. Clients (replication of the <i>Live Server</i> , publication, etc.) remain paused. Appropriate messages are written to the server log.
Error correction	As soon as the database is available again the server creates new connections to the database (watch the server log) and blocked transactions are released. Note: Error-free operation after a database breakdown cannot be guaranteed, since external components are also affected (for example the JDBC driver). If an error occurs here, it is usually recognized by the watchdog and remedied by restarting the server.

Table 2.3. Database failure

Malfunctions of the Replication Live Server

Malfunctions that occur on the *Replication Live Server* can affect two components:

- replication: Replication is interrupted and the content of the *Replication Live Server* becomes outdated depending on the publication activity of the *Content Management Server*. After restart, the *Replication Live Server* continues replication at precisely the point where it was interrupted and will only go online when its content is up to date.
- *Content Application Engines*: Not running *CAEs* fail to start and running *CAEs* return errors when requesting the *Replication Live Server*. After restart, *CAEs* can reconnect. They don't need to be restarted.

Replication Live Server Failure	
Error behavior	All CAEs connected to the <i>Replication Live Server</i> receive connection error responses from the server.
Error correction	The watchdog of the <i>Replication Live Server</i> detects the failure and restarts the server (see the watchdog and <i>Replication Live Server</i> logs). As soon as the <i>Replication Live Server</i> is online again, the CAEs are working properly again.

Table 2.4. *Slave Live Server failure*

Replication Live Server Database Failure	
Error behavior	Transactions which are active at the time of failure or which first notice the failure are terminated with an error. The error is passed to the server and clients. A replicator client is terminated. Requests to CAE clients fail with an error. Transactions started after the server has detected the database failure are blocked until a new database connection is created. Requests to CAEs are paused. Appropriate messages are written to the server log.
Error correction	Because a database failure can lead to erroneously generated pages or blocked requests, the watchdog restarts the <i>Replication Live Server</i> . In the meantime the second <i>Replication Live Server</i> takes over the task of the first one. Restarting the <i>Replication Live Server</i> implies a brief process failure. In its initialization phase, the <i>Replication Live Server</i> will remain paused, until the database is available again (see server log). The CAEs cannot log in this state.

Table 2.5. *Database failure*

Malfunctions of the CAE

A Failure in the CAE directly affects web page generation with JSPs. This applies to non cacheable content as well as cacheable content generated for the first time.

CAE Failure	
Error behavior	Requests which were active at the time of failure are canceled. Later requests are refused. In both cases a browser displays an error page.
Error correction	In the cluster system, the load balancer must remove the failed CAE from its list of active web servers.

Table 2.6. *CAE failure*

CAE Deadlock	
Error behavior	Request are blocked and a browser displays an error.
Error correction	The watchdog of the CAE detects the error and restarts the CAE (see the watchdog log).

Table 2.7. CAE deadlock

2.3 Multi-Site Publishing

Overview

If you are using *Multi-Site Management* for a *CoreMedia CMS* installation, more than one *Master Live Server* may be connected to one *Content Management Server*. To each of the *Master Live Servers* a number of base folders is assigned, that is, folders located immediately below the root folder of the *Content Management Server*. When performing a publication, the target *Master Live Server* is determined by the base folder that contains the resources to be published.

In this scenario, each base folder can be thought of as an isolated single-site *Content Management Server*. Resources cannot be moved across boundaries of base folders, just as they cannot be moved outside a *Content Management Server*. Publications can only span single base folders and published links must always connect resources within one base folder. This ensures that a base folder on the *Live Server* can be guaranteed to be self-contained, avoiding dangling link problems during delivery.

In the following you will get some hints on how to deploy *Multi-Site Management* in different contexts.

Multiple Web Sites

Use Cases

Situation: You want to produce content for multiple websites. Due to load balancing or stability constraints, the sites must be hosted on separate servers.

Solution: You create one top-level folder for every website and install one *Master Live Server* for every website. You map folders and targets one-to-one. You assign rights to editors as needed, possibly allowing access to more than one site for certain users.

Result: By sharing the *Content Management Environment* you reduce the hardware requirements. You can more easily administrate the multiple sites in a common framework. Editors can log in once and gain access to all sites for which they are authorized.

Intranet/Internet

Situation: You want to produce content for a company's intranet and for the Internet on one *CoreMedia CMS* installation. As intranet and Internet servers have different security requirements, they must be located on physically separated machines each on the correct sides of a firewall.

Solution: You create base folders `internet` and `intranet` and map each of them to one publication target using the two different servers. Users are mapped into the *Content Management Server* from the company's LDAP server.

Result: Similar to the multi-website scenario, but you get the additional benefit of a secure deployment by introducing a firewall.

Lots of Little Sites

Situation: You host a number of websites, none of which is particularly large or under heavy load.

Solution: You create one base folder per site, but only a single publication target. All base folders are mapped to that target. A URL rewriter in front of the CAE maps each site to one top-level folder.

Result: Link and move barriers ensure that for each site there are no internal links that accidentally leave the site. As the number of little sites grows, you may later map additional sites to new publication targets without having to rework the old sites.

2.4 Server Run Levels

The *CoreMedia Content Server* has several modes of operation (or run levels). These run levels determine which programs can connect to the server and which operations can be executed on the server. Each mode has the abilities of the previous mode (maintenance<administration<online). The following modes are available:

offline	The state when the server is not running and no operations can be executed.
maintenance	The server is accessible for maintenance or debugging purposes. Only the standard server tools can be used.
administration	The server can be managed by JMX. The publisher user may log in. In the case of a <i>Replication Live Server</i> , replication starts.
online	All services are active (see below). All users and clients can log in.

The following services are active in the various server run levels:

maintenance	CORBA Communication License Manager Login for "admin" and "debug" services Access to resources
administration	Replicator (<i>on Replication Live Servers</i>) Login for Publisher
online	<i>Content Feeder</i> Login for all other clients and programs, like Editor, Importer and File System View

There are two possibilities to reach a particular run level:

Start in a particular run level

You can use the property `cap.server.init.runlevel` in `contentserver.properties` to define the run level that the *Content Server* should reach on start up.

Allowed run levels are `maintenance`, `administration` and `online`. The default level is `online`.

Switch to a particular run level

To switch a run level in a running server, execute the command:

```
cm runlevel -u admin -r <runlevel> -g <grace period>
```

After the `<grace period>` (in seconds), the server changes to the run level given by `<runlevel>`.

During the grace period, you can cancel the run level switch with the following command:

```
cm runlevel -u admin -a
```

Clients logged on the server are informed at regular intervals when they will be logged out due to a "down" switch.

The *Site Manager* warns users with a dialog box when the server shuts down.

Shut down the server with the command `cm runlevel -r offline -g <grace period>`.

2.5 Changelog

The *CoreMedia CMS* is event driven. For each repository change an event is generated and send to all listeners. Listeners are for example:

- The Replicator of a *Replication Live Server*
- The *Workflow Server*
- The *Content Feeder*

It is also possible to write own clients which act as listeners.

All events are written to the `ChangeLog` table in the database with a timestamp attached. Therefore, a listener which was offline for a while can catch up with the current repository state by replaying all events since it was offline.

3. Configuration and Operation

This chapter describes tasks necessary for the administration and configuration of the *CoreMedia Content Servers*. This chapter might not cover all possible tasks. If you are missing some tasks, you should have a look in [Chapter 5, Appendix \[226\]](#). There you will find a description of all relevant properties, that you might use for configuration of the *CoreMedia CMS*.

3.1 Structure of Content Server Installation

When you install the *Content Server* into the servlet container, it will have the following file structure:

```

coremedia.war
├── WEB-INF
│   ├── classes
│   ├── config
│   └── contentserver
│       ├── doctypes
│       └── spring
│   ├── lib
│   ├── properties
│   └── corem
│       ├── application.properties
│       ├── application.xml
│       └── logback.xml

```

The `config` directory contains the Spring configuration and the content type definitions. The `properties/corem` directory contains the main configuration. The `application.xml` adds the site model to the Content Server.

For detailed information about the architecture of CoreMedia applications, please have a look at Section 4.1.3, “Application Architecture” in *CoreMedia Digital Experience Platform 8 Developer Manual*.

Section 4.3.8, “Managing Properties in the Workspace” in *CoreMedia Digital Experience Platform 8 Developer Manual* describes how properties can be overridden.

Web application structure

Example 3.1. Structure of the Content Server web application

3.2 Deploying the Content Server

The *Content Server* is delivered as a standard WAR archive and thus can be deployed to a servlet container like any other web application. The *Server Tools* are not bundled with the *Content Server* and have to be installed separately.

Web application deployment

Although it is not recommended due to performance and resource issues, it is possible to deploy all *Content Servers* to a single servlet container, for a test scenario, for instance. If you do so, be aware of the following issues:

Deploying multiple servers to a single container

→ *Context Path*

Each *Content Server* web application needs a unique context path. Therefore, all clients that connect to this server must have the context path in its connection URL. That is, for example, the URL defined in `cap.client.server.iior.url` cannot have the context "coremedia" for all servers.

→ *Startup issues*

You should configure the servlet container to not accept requests during startup. For Tomcat this can be set with `bindOnInit="false"` in the connector element of the `conf/server.xml` file. Otherwise, it may happen that some *Content Server* waits for another one which appears later in the servlet container's startup sequence (for example, the *Content Management Server* tries to connect to the *Master Live Server* on startup, but Tomcat happens to start the *Content Management Server* first). Setting the `bindOnInit` flag helps to recognize the situation early, without waiting for a timeout.

3.3 Configuring the Database

Please read the Supported Operating Environments document on the *CoreMedia DXP 8* documentation [page](#) in order to get the actually supported databases.



The configuration of the database is done using the `sql.properties` file and the database dependent `<DBName>.properties` files. The *Content Server* connects with the database using JDBC. The JDBC driver JAR files must be installed in the `lib` directory of the *CoreMedia CMS* installation.

Note: The required JDBC drivers are not part of the *CoreMedia CMS* distribution; they must be obtained from the corresponding database manufacturer.



The database must be accessible over the network and should be enabled for the Unicode character set. Each *Content Server* requires a different database user account. The database users must be able to create and delete tables and indexes in their schemas. (For Oracle, see [Example 3.2, “Oracle: create DB User” \[35\]](#) and [Example 3.4, “Oracle: delete DB User” \[35\]](#))

Manual creation of tables is not necessary. A *Content Server* automatically creates tables, the first time it is started. The standard settings of the database or the standard values defined for the database user are used as default for the tables and indexes created, such as the initial reserved memory.

Important: To guarantee high performance of the whole system, correct installation and regular database maintenance is necessary. In particular, the statistics data for tables and indexes must be updated daily according to the manufacturer's instructions.



3.3.1 Specifying Tablespaces for the Content Server

In general, the *Content Server* creates default database tables in default table spaces but it's also possible to define custom table spaces for specific content categories.

Configuring a table space for blob data

The binary content of blobs stored in the *CoreMedia CMS* has different access characteristics than other content properties: it usually read from beginning to end as a stream, and blobs are generally too large to be cached in database main memory. Therefore, you may want to store blob data in a different table space than other tables automatically created by *CoreMedia CMS*.

The required settings depend on the type of database in use and are described in this chapter for all supported databases. The settings described are only basic

configuration steps. When tuning is required, a database administrator has to be involved.

Generally, the *Content Server* creates the table holding blob data when you start it first. You can augment the DDL create statement used by the *Content Server's* SQL scripts with own extensions. Use the property `db.blob.tableoptions` in the database specific property file (`properties/corem/db2.properties`, for example).

By default, this property is commented out, causing the database to select a default table space. The commented line shows the syntax of the relevant SQL fragment for the respective database.

The property `db.blob.tableoptions` only takes effect on the first startup of the content server.



DB2

The supplied SQL fragment shown below is appended at the end of the CREATE TABLE statement. Its syntax defines a table space for the blobs metadata (here `myBlobMetadataTablespace`), and a table space for the actual blob data (here `myBlobTablespace`). Since the metadata is small, you can usually use the database's default table space for the metadata.

```
db.blob.tableoptions=IN myBlobMetadataTablespace  
LONG IN myBlobTablespace
```

Oracle

The supplied SQL fragment is appended in the LOB STORE AS clause. The example syntax below refers to a table space `myBlobTablespace` for holding the actual blob data.

```
db.blob.tableoptions=TABLESPACE myBlobTablespace
```

MS SQL

The supplied SQL fragment is appended at the end of the CREATE TABLE statement. The example syntax below refers to a table space `myBlobTablespace` for holding the actual blob data.

```
db.blob.tableoptions=TEXTIMAGE_ON myBlobTablespace
```

Configuring tablespaces and index tablespace for content tables

Content type specific tables vary considerably in their size and access frequency. Therefore, it might make sense to distribute them across different table spaces.

Also, databases allow you to store indexes in a different table space than the original data.

The SQL DDL statements for content type specific tables are created automatically by the *Content Server*. To control these statements, set the attributes `Tablespace` and `IndexTablespace` in your `doctype.xml` file.

Selecting a table space per content type

To select the table space for a content type specific table, set the attribute `Tablespace` on the respective content type's definition. The example uses a table space `bigTablespace` for the tables of content type `Article`.

```
<DocType Name="Article" Tablespace="bigTablespace">
  ...
</DocType>
```

Selecting an index table space per content type

To select the table space for a content type's primary key index and auxiliary indexes created by the *Content Server*, set the attribute `IndexTablespace` on the respective content type's definition. This also configures the default table space for all indexes created on content properties of the respective content type. The example uses a table space `quickTablespace` for the index tables of the content type `Article`.

```
<DocType Name="Article" IndexTablespace="quickTablespace">
  ...
</DocType>
```

Selecting an index table space per property

To select a table space for an index on an indexed content type property, set the attribute `IndexTablespace` on the respective property's definition. This is only possible for String, Integer, and Date properties, but not for XML, blob and link list properties. The example uses the table space `quickTablespace` for the index tables of the property `ArticleCode` of the content type `Article`.

```
<DocType Name="Article">
  <StringProperty Name="ArticleCode" Length="20" Index="true"
    IndexTablespace="quickTablespace"/>
  ...
</DocType>
```

The "index table space per property" feature is not supported by DB2 databases.



3.3.2 Oracle Database

For the Oracle database you need the Oracle JDBC Thin Driver `oracle.jdbc.driver.OracleDriver`. Copy the driver file to the `WEB-INF/lib` directory. There also exists the Oracle OCI driver which could be used. However, this driver can cause some problems:

- If the database crashes, the OCI driver hangs/blocks and the *Content Server* must be restarted.
Connections can hang during rollback.
- Problems with the streaming of blobs can occur.
- Oracle recommends the thin driver for performance reasons.

Furthermore, the driver needs an additional Oracle client installation on the *Content Server* host. Therefore, the OCI driver is not recommended.

Important: Always use the latest version of the JDBC driver for your database version and the JDK being used. New driver versions can be obtained from the Oracle website <http://technet.oracle.com>.

In order to optimize queries, Oracle gathers statistics about the database content. By default, this process runs once per day, which is an appropriate setting.

In `sql.properties`, the following settings must be made for an Oracle database:

```
sql.store.driver=oracle.jdbc.driver.OracleDriver
sql.store.url=jdbc:oracle:thin:@<DB-HOST>:<DB-PORT>:<DB-INSTANCE>
sql.store.user=<DB-USER>
sql.store.password=<DB-USER-PASSWORD>
```

The name of the database user may be given in lowercase in the `sql.properties`, but it must be stored as all uppercase in the database.

The Oracle database instance must be configured with a sufficient number of *DB Cursors*. The number is set in the Oracle initialization script `init<Instance name>.ora` in the entry `open_cursors` (for further details see the documentation of the database manufacturer).

The number of cursors for a JDBC connection in the *CoreMedia Server* is calculated with the following formula:

```
<Number of Cursors> = 110 + (<Number of content types> * 7)
```

For a server with 15 content types therefore, 215 cursors are needed per connection and an `open_cursors` value of 215 has to be configured, unless other applications require a higher value.

Alternatively the property `sql.store.preparedStatementCacheSize` in the `sql.properties` can be used to control the number of open cursors from the *Content Server*. This property limits the number of cached prepared SQL statements. If the property is not set, the cache has an unlimited size and the above formula applies. Otherwise, the value of the property should be at least 50. Increasing this parameter generally improves the database performance. Set the number of Oracle cursor to the size of the statement cache plus 10 for other statements that are not prepared, but that still use a cursor when executed.

```
create user <DB-USER> identified by <DB-USER-PASSWORD>;
grant connect, create view, resource to <DB-USER>;
```

*Example 3.2. Oracle:
create DB User*

For Oracle 12c you need to add privileges to the user as described in http://docs.oracle.com/database/121/DBSEG/release_changes.htm#BABEBGDI.

```
alter user <DB-USER> quota unlimited on users;
```

*Example 3.3. Add
privileges for Oracle
12c user*

```
drop user <DB-USER> cascade;
```

*Example 3.4. Oracle:
delete DB User*

Optimization Options

In order to optimize your ORACLE database, you should consider the following hints:

- Use an I/O subsystem with write caches.
- Five hard disks are recommended for database operation: one each for redo logs, rollback, index, data, and archive logs.
- Striping increases performance further. Hard disks could be mirrored in pairs, and striping could be defined across the mirror pairs.
- RAID 5 should be avoided for performance reasons.
- Larger redo logs might be necessary (for example 500MB). The aim is to rotate the logs less than once every three minutes. Larger redo logs increase recovery time.
- Large redo log buffers (such as 10 MB) are recommended.
- The tables `Blobdata` and `PK_Blobdata` should have larger than default values of `init_trans`, for example, 10 or 20 (depending on the number of concurrently active database connections, which in turn is defined in the `sql.properties` file of the *Content Management Server*). The parameter `init_trans` determines how many processes can write to a single block in parallel. Large "waits for data block" in the `statspack` report indicate an `init_trans` problem. Increasing `init_trans` by 1 requires about 25 additional bytes per block for management information.

- ➔ To check if `init_trans` is too small, this command shows the waiting database jobs: `select sid, sql_text from v$session s, v$sql text t where s.sql_hash_value=t.hash_value and s.sid in (select sid from v$session_wait where event = 'enqueue') order by sid, piece;`
- ➔ Be sure to update the database optimizer statistics on a regular basis

3.3.3 IBM DB2 Database

For DB2 database configuration look into the CoreMedia IBM Deployment Manual.

3.3.4 Microsoft SQL Server

For the SQL Server database you must use the original JDBC driver supplied by Microsoft. Copy the driver file into the `WEB-INF/lib` directory and configure the following settings in the file `sql.properties`:

```
sql.store.driver=
  com.microsoft.sqlserver.jdbc.SQLServerDriver
sql.store.url=jdbc:sqlserver://
  <DB-HOST>:<DB-PORT>;databaseName=<DB-NAME>
sql.store.user=<DB-USER>
sql.store.password=<DB-USER-PASSWORD>
```

Only SQL Server authentication is supported for the database user.

The DB user must not be a DBA or owner of the database otherwise all objects are created in the wrong (`dba`) schema. Roles `db_datareader`, `db_datawriter` and `db_ddladmin` are sufficient as the rights for the database user.



3.3.5 PostgreSQL Database

PostgreSQL is an open source SQL database available from <http://www.postgresql.org>. For the certified PostgreSQL database you need the following JDBC driver:

- ➔ Up-to-date PostgreSQL JDBC4 driver for your version of PostgreSQL

The JDBC driver is available from the URL <http://jdbc.postgresql.org>.

Configure the following settings in the `sql.properties` file:

```
sql.store.driver=org.postgresql.Driver
sql.store.url=jdbc:postgresql://<host>:<port>/<databasename>
sql.store.user=<DB-UserName>
sql.store.password=<DB-UserPassword>
```


In a standard PostgreSQL installation the port is 5432. So replace `<port>` with 5432 and `<host>` with the name of your PostgreSQL computer. Replace `<data basename>` with the name of the PostgreSQL database you created using the `createdb` command.

When creating the database, take care to select Unicode character encoding:

```
createdb -E UNICODE <my-databasename>
```



You need to create a schema for each database user, because *CoreMedia CMS* requires that the schema name equals the user name. Otherwise, all tables would be created in the public schema. You also have to create the schema before starting the *Content Server* for the first time. For example, the following commands can be used at the `psql` command line in order to create one database user for the *Content Management Server* and *Workflow Server* (named `cm_mgmt` in the example) and one database user for the *Master Live Server* (named `cm_master` in the example):

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;  
CREATE USER cm_mgmt PASSWORD 'secret';  
CREATE SCHEMA cm_MGMT AUTHORIZATION cm_mgmt;  
CREATE USER cm_master PASSWORD 'topsecret';  
CREATE SCHEMA cm_master AUTHORIZATION cm_master;
```

It is your choice whether you use different schemas in one database, or separate databases. Both deployment variants are supported.

PostgreSQL requires regular maintenance for proper operation. Please see the PostgreSQL manual for `VACUUM` and `ANALYZE`. You may also want to use **pg-autovacuum**.



Storing blobs in PostgreSQL largeObjects

On PostgreSQL, a different storage format inside the database is used for large blobs. The native PostgreSQL largeObject format supports streaming, and will not cause the server to run out of memory on huge blobs, in contrast to PostgreSQL's `bytea` type used before.

By default, `bytea` is used for blobs up to 1 MB, largeObject is used for larger blobs. Note that largeObject storage has administrative implications:

- If you use `pgdump` for a complete data base backup, you may use different formats than the standard plain text format due to size and speed. For example, use `-Fc` if you have a lot of binary data. If you want to do a selective backup (schema or table), you should check the PostgreSQL documentation on <http://www.postgresql.org> for details.
- Access control for blobs in the DB is only possible on a per-database level, not per schema.

- `cm schemaaccess dropAll` will work in the sense that the data base appears empty to the content server process, but it will not delete the Blobs and therefore does not free the hard disk space.

3.3.6 MySQL Database

The MySQL Connector/J provides JDBC support for MySQL. Please use the version 5.1.35 or later. The driver is available from <http://dev.mysql.com/downloads/connector/j/5.1.html>.

Copy the driver file to the `lib` directory of the *Content Server* installations. Afterwards, you can configure the database in the file `properties/corem/sql.properties` as follows:

```
sql.store.driver=com.mysql.jdbc.Driver
sql.store.url=jdbc:mysql://localhost:3306/<user>
sql.store.user=<user>
sql.store.password=<password>
```

Replace `<user>` and `<password>` as appropriate. Lower case is recommended. You create a separate database and a separate user for each server as follows:

```
CREATE DATABASE <user> CHARACTER SET = 'utf8mb4'
COLLATE = 'utf8mb4_bin';
GRANT ALL PRIVILEGES ON <user>.* TO '<user>'@'%'
IDENTIFIED BY '<password>';
```

Again, replace `<user>` and `<password>`. If not properly set as indicated above, the *Content Server* will change the character set and the collation of the database to `utf8mb4` and `utf8mb4_bin` during the first start.

All tables will be set up to use the InnoDB storage engine. By default, the caches for this storage engine are configured very small. Consider increasing the MySQL startup option `innodb_buffer_pool_size`. Also, increasing `innodb_log_file_size` will improve write performance, because log files are rotated less often.

3.4 Configuring Blob Storage

Up to *CoreMedia CMS 2006*, the *Content Server* stored all blobs in one database table. This was sufficient for many cases, but there are scenarios where other solutions are preferable, for example:

- Different access characteristics for different blobs
- Very high volume of blobs
- Requirements on upload and delivery speed

CoreMedia CMS now offers two methods to cope with different requirements for blob storage:

- Configure the table space for blob data as described in [Section 3.3.1, “Specifying Tablespaces for the Content Server” \[31\]](#)
- Define different "media stores" for blobs, described in this section.

Media Stores for blobs

CoreMedia CMS supports different media stores for the storage of blobs, depending on some properties of the blob, such as

- size,
- MIME type
- target property and content type.

The Media Store is only responsible for storing and retrieving the actual bytes. The MIME content type, garbage collection, and authentication is managed by the *Content Server*. *CoreMedia CMS* supports the following media stores:

- Storing blobs in files
- Storing blobs as ZIP files
- Zip blobs and store them in the `blobstore` table
- Storing blobs in different database tables

By default, blobs are still stored in the *CoreMedia CMS*.

File-based blob storage

As one option for a blob store, you can store blob contents on the file system reachable by the *Content Server*. The *Content Server* takes care that the number of

files/subdirectories per directory does not grow too large. By default, files are stored in the `blobstore/file` directory below the *Content Server* installation.

For consistent backup, suspend the server's blob collector with the property `sql.store.collector.suspend=true` in the `sql.properties` file before starting the backup process.

This feature is not intended to manage existing file structures, and does not support file access that bypasses the *Content Server*. Also, a file structure may be accessed by only one *Content Server* at a time.



Storing blobs as zipped files

You can compress blobs which are not compressed by themselves, such as text blobs, and store them in the file system or in the database. By default, the zipped files are stored in the `blobstore/zipfile` directory below the *Content Server* installation directory or in the `blobstore` table of the database. Storing zipped files in the database requires a temporary directory which is `var/temp` by default.

Storing blobs in different database tables

You can configure multiple database tables, each with individual database-specific characteristics, to store the blobs in. The *Content Server* will not create the tables automatically, so you have to create them manually.

What happens to existing blobs

If you add a blob store for an existing *Content Server* chances are high that blobs are already stored in the database. These blobs will not automatically be moved to the newly configured blob store. On the *Master Live Server*, the blobs will be moved to the new store the next time when the content item containing the blob is published. On the *Content Management Server* you have to download the blob from the content item and upload it again it's not enough to create a new content item version.

Configuring blob store selectors

Not Mandatory: You only need to do this configuration, if you want to store your blobs in a location different from the default location in the CMS.



Configure the blob stores in the file:

→ `blobstore<yourName>.xml` in the `config/contentserver/spring/blobstore` directory.



Replace "*<yourName>*" with a name of your choice; The *Content Server* loads all XML files from the directory and adds them to the default blob store definitions.

Take the `blobstore-example.xml` file as an example and see the Javadoc of the `hox.corem.server.media` package for further details.

If you want to store blobs in different locations, depending on some properties of the blob, you can add media store selectors to the *Content Server*. A media-store selector defines under which conditions a specific blob store should be used. [Example 3.5, "Example configuration" \[41\]](#) shows the configuration of two media store selectors. One for blobs larger than 1000000 bytes, which are stored in the file system, and one for images smaller than 8192 bytes, which are stored in a different database table.

Example 3.5. Example configuration

```
<customize:prepend bean="blobstore"
  property="mediaStoreSelectors">
<description> Select a media store </description>
<list>
  <!-- store big blobs in the file system. -->
  <bean class=
    "hox.corem.server.media.ConditionalMediaStoreSelector">
    <property name="storeName" value="file"/>
    <property name="condition">
      <bean class="hox.corem.server.media.MatchCondition">
        <property name="minimumLength" value="1000000"/>
      </bean>
    </property>
  </bean>

  <!-- store images of a certain minimum size in a different
  database table -->
  <bean
    class="hox.corem.server.media.ConditionalMediaStoreSelector">

    <property name="storeName" value="dbblob"/>
    <property name="condition">
      <bean class="hox.corem.server.media.MatchCondition">
        <property name="primaryType" value="image"/>
        <property name="maximumLength" value="8192"/>
      </bean>
    </property>
  </bean>
</list>
</customize:prepend>
```

The following blob store names are predefined:

- ➔ "file" for storing blob contents in the file system
- ➔ "dbblob" for storing blob contents in the database table `blobstore`. Consult the Javadoc of `hox.corem.server.media.BlobStore` for further details.
- ➔ "zipfile" for storing blob contents in the file system in zipped format

- ➔ "zipdbblob" for storing blob contents in the database table `blobstore`, in zipped format.
- ➔ the empty string, "", for storing blob contents in the CMS tables as in previous releases.

Media store selectors are stored in the property `mediaStoreSelectors` of the `blobstore` bean. You can use the customizer tag `<customize:prepend>` to prepend your media store selectors to the list. The *Content Server* iterates over this list and the first matching condition defines the blob store to use for the specific blob.

If you need to, you can use the full power of Spring for configuration. However, as an administrator, you only need the following beans and tags to construct your condition.

The `ConditionalMediaStoreSelector` is the bean which chooses the blob store to use. It has a property `condition` which takes the condition.

<bean>

Child elements: `<property>`

Parent elements: `<list>`, `<customize:prepend>`, `<customize:append>`

The `<bean>` element defines the bean which should be configured. In the context of the blob storage media store class or condition bean, which should be configured.

Property	Description
class	Defines the Bean class, which should be modified. In the context of the blob storage, these are media store beans or condition bean, which should be configured.

Table 3.1. Attributes of the bean element

The following condition beans can be used.

Condition Name	Description
<code>hox.corem.server.media.AndCondition</code>	A condition which takes a list of conditions in the <code>conditions</code> property. You configure a list with the <code><list></code> element. It returns "true" when all conditions are fulfilled.
<code>hox.corem.server.media.OrCondition</code>	A condition which takes a list of conditions in the <code>conditions</code> property. You configure a list with the <code><list></code> element. It returns

Table 3.2. Condition classes which could be used in the `<bean>` element.

Condition Name	Description
hox.corem.server.media.Match Condition	<p>"true" when at least one of the conditions is fulfilled.</p> <p>A condition which takes some values defined with <code><property></code> elements and which compares these values with the values of the blob to store. The MatchCondition can take the following values:</p> <ul style="list-style-type: none"> → <i>primaryType</i> - Required primary MIME type of the content. → <i>subType</i> - Required sub MIME type of the content. → <i>typeParam</i> - Required parameter for the MIME type of the content. Can be either of the form "name=value" to test for a specific value for a type parameter, or of the form "name" (without '=') to test for presence of a "name" parameter. → <i>minimumLength</i> - Required minimum content length in bytes. → <i>maximumLength</i> - Required maximum content length in bytes. → <i>documentTypeName</i> - Required content type name of the target content item. → <i>propertyName</i> - Required property name of the target property. → <i>isVersion</i> - Required property of the version attribute. That is, will the blob be stored as a version ("true") or as content. <p>It returns "true" when all values match.</p>

<property>

Child elements: `<bean>`, `<list>`

Parent elements: `<bean>`

The `<property>` element defines the property of a bean, which should be configured.

Table 3.3. Attributes of the property element

Attribute	Description
name	The name of the bean property, which should be configured.
value	The value, which should be added to the bean property.

<list>

Child elements: <bean>

Parent elements: <property>

The <list> element groups conditions, which should be used in an `AndCondition` or `OrCondition`. It has no attributes.

Proceed as follows to configure the `ConditionalMediaStoreSelector`:

1. Take the `blobstore-example.xml` file as a template for your media store selector.
2. Add a new `<bean class="hox.corem.server.media.ConditionalMediaStoreSelector">` tag below the `<list>` tag of the `<customize:prepend bean="blobstore" property="mediaStoreSelectors">` tag or uncomment an existing configuration from the file.
3. Add a `<property name="storeName" value="selectedMediaStore"/>` below the `<bean>` tag and replace "selectedMediaStore" with the name of the media store where the blob should be stored.
4. Construct your matching condition.

Configuring store locations for file storage

You can change the default locations for the storage of blobs in the file system.

1. Open the file `blobstore.properties` in the `config/contentserver/spring/blobstore` directory.
2. Uncomment the appropriate property and add your path:
 - For the location of blobs stored in the file system use the `cap.server.blobstore.file.rootdir` property.
 - For the location of blobs zipped and stored in the file system use the `cap.server.blobstore.zipfile.rootdir` property.
 - For the location of temporary files for blobs zipped and stored in the database use the `cap.server.blobstore.zipdbblob.tmpdir` property.

3.5 Exclusive Locks

The server, the publisher, and the replicator require exclusive database transactions, but only during their startup phases. After the server components are up, some operations will still block all concurrent writes. Because the server executes requests in a first come, first served fashion, this might also block read requests that reach the server when a write is pending. Therefore, you should use such operations only when necessary.

The following operations block concurrent writes:

- Creating, dropping, or refreshing the folder index using the `cm dbindex` tool. If possible, use this tool in times of low load.
- Cleaning the recycle bin using the `cm cleanrecyclebin` tool. If possible, use this tool in times of low load.
- Fetching a timestamp using the method `ContentRepository.getTimestamp()`. Occasionally this might be necessary to bootstrap a repository listener, but in general you should use this method sparingly.
- Providing a synthetic replay for a repository listener. This will generate a sequence of events that might have led to the current repository state. A synthetic replay is generated by adding a content repository listener with the timestamp `Timestamp.SYNTHETIC_REPLAY`. Whenever possible, you should use `ContentRepository.getContents()` to access all contents.

3.6 Configuring CORBA

Clients communicate with the *Content Server* using the CORBA protocol. To this end, each client must keep open a TCP connection through which it passes commands to the server and receives operation results and events from the server. By default, the Oracle ORB provides resources for up to 240 connections.

If you expect connections from more than 230 concurrent JVMs, you should set the ORB's TCP connection pool size to the number of concurrent clients plus 10, granting some headroom. To this end, you have to set the system property `com.sun.CORBA.connection.ORBHighWaterMark` to the desired numeric value.

If you are unsure about the maximum number of concurrent clients, use the number of concurrent logins specified in your license as a safe approximation.

3.7 Extending the Content Server

The *Content Server* provides extension points to modify its behavior.

Extending the Publisher

The publisher provides extension points to modify published contents on the fly. You might for example prevent certain blob properties of a document from being transferred to the *Master Live Server*. To do so, implement the interface `PublishInterceptor` or better extend `PublishInterceptorBase` and add the interceptor bean to the application context. It will be automatically called on publication if the interceptor declares to be applicable for the given content type.

One example for such an interceptor is available for *CoreMedia Asset Management*. Have a look at Section “Configure Rendition Publication” in *CoreMedia Digital Experience Platform 8 Developer Manual* for more information.

You must not (and you cannot) modify any internal links during publication. This includes links within markup. Trying to do so will fail the publication with an exception.



3.8 Starting the Server

What happens when you start the server depends on whether you start the server for the first time after installation or if you start it later again

First CM Server start

Before you start the *Content Server* for the first time, make sure that the content type definition file (`<xyz>-doctype.xml`) can be found at the location configured with the parameter `cap.server.documentTypes` in the file `contentserver.properties`. Normally this file is placed in the directory `config/contentserver/doctype`s. You may point this parameter to multiple comma separated paths, which may contain Ant-style wildcards. If you want to use separate table spaces for blob data, you have to adapt the `<DBname>.properties` file of your database before the first start of the *Content Server*. See [Section 3.3.1, “Specifying Tablespace for the Content Server” \[31\]](#) for details.

Start the servlet container that contains the *Content Server* as described in the servlet container documentation.

When executing for the first time, this creates the data schema from the content type definition files. Depending on the file number and sizes this can take some time. Look into the log file of the *Content Server* and the `catalina.out` to see system and error messages.

Make sure that the *Master Live Server* is online before you start any *Replication Live Servers*. After the first start of a *Replication Live Server*, the server will remain in the runlevel `Administration` after the content of the *Master Live Server* has been completely replicated. Run `cm runlevel -u admin -r online -g 0` to complete the setup of the server, possibly after verifying that the replication completed without errors (see [Section 3.10.1, “Installing the First Replication Live Server” \[54\]](#) for details). Make sure to create the necessary users and groups with the appropriate membership relations.

Start the servlet container that contains the *Workflow Server* as described in the servlet container documentation.

Now you need to choose and upload predefined workflows to the *CoreMedia Workflow Server* that you want to use. You can omit this step if you want to use only your own workflows with your own workflow groups. The valid predefined workflow names are defined in the [Workflow Manual]. To upload one of these predefined workflows with the `upload` utility, execute `cm upload -u admin -p <password> -n <workflow-name>` and replace `<password>` with the password of the admin user (ID=0) and `<workflow-name>` with one of the names listed in [Workflow Manual], for example, `three-step-publication.xml`.

Prerequisites

*Start web application
Content Management
Server
Check for proper start-up*

*Start the Replication
Live Server*

*Start the Workflow
Server*

Upload workflows

You can continue uploading further custom workflows in the same way as described for the Global Search and Replace workflow in the previous step. The upload utility is described in the [Workflow Manual].

After successful initialization, the password of the administrator (admin) and other system users (watchdog, importer, etc.) should be changed in order to guarantee the security of the system. Go to the User Management window in the editor for this purpose (see [Section 3.16, “User Administration” \[160\]](#)). Make sure to update all relevant configuration files.

Later Content Server starts

Later starts of the *Content Server* with the servlet container will be faster.

A *Replication Live Server* will go online only after it has contacted the *Master Live Server* and has replicated all but a few very recent changes. This means that the server will not go online automatically if the replicator is disabled or if the *Master Live Server* is not running.

Wait until the server is up and start the *Workflow Server*.

For description of the options see [Section 2.4, “Server Run Levels” \[25\]](#).

The server can be stopped with

```
cm runlevel -r offline [-g<grace period>] or with the servlet container's methods.
```

Running clients are informed in regular intervals about the server shutdown.

To check whether the *Content Server* is running, try to reach the server, for example with Telnet:

```
telnet <serverhost> <server.http.port>
```

If the server is running, a connection can be established, otherwise it fails.

Start both servers before all other *CoreMedia* components which must communicate with them. A server restart is necessary when the database connected to the server is restarted.

3.9 Recovery of Content Server Databases

The following chapter provides an overview of the recovery scenarios for the *Content Server's* databases - the databases of the *Content Management Server* and the *Master Live Server*. Recovery of the *Replication Live Server* is described in [Section 3.10, “Administrating Replication Live Servers” \[54\]](#) but you should also read [Section 3.9.1, “Backup Strategy” \[50\]](#). The recovery strategy depends on the server whose database has been corrupted.

3.9.1 Backup Strategy

You need to have database backups to recover from database failures. The backups are created with database tools. The chronological order of the backups is crucial:

1. Backup one of the *Replication Live Servers*.
2. Backup the *Master Live Server*.
3. Backup the *Content Management Server*.

Note, that recovery will not work correctly, if this given chronological order of backups is not respected. The content of the *Content Management Server* must be newer than the content of the *Master Live Server*, and the content of the *Master Live Server* must be newer than the content of the *Replication Live Servers*. The time between the single backups should be short. Try to avoid publishing deletions while the backups are made.



When you backup the database of any server, make sure that the database is consistent, that is, it represents the exact state of the database at a certain point of time. The exact backup procedure depends on your database product and likely on the configuration of your database.

If you use a custom blob store as described in [Section 3.4, “Configuring Blob Storage” \[39\]](#) and if the blob data is not stored in the same database as the remainder of the content data, use the following procedure for each server:

1. Set the property `sql.store.collector.suspend` of the file `properties/corem/sql.properties` to `true`.
2. Wait 20 seconds to give the blob collector a chance to shut down.
3. Backup the *Content Server* database.
4. Backup the data store. This might be a file system backup or a backup of a separate database.
5. Set the property `sql.store.collector.suspend` to `false`.

If the collector is suspended, it will collect blobs again when it is run the next time. By default, it runs once a day. This might be a problem, if your backup procedure also mandates daily backups, continually preventing the collection of blobs. If you see the message "Blob deletion is suspended" with log level info on the log facility `hox.corem.server.sql.SQLStore.blobcollector` more than occasionally, you should set the property `sql.store.collector.startTime` in such a way that the collector starts soon after your backup is finished. The property `sql.store.collector.startTime` is given in seconds after the start of the day in the default time zone.

3.9.2 Recovery of a Content Management Server Database

If the database of a *Content Management Server* is corrupted, proceed as follows:

1. Stop the *Content Management Server*. The sessions of the connected clients will be closed and no more content changes are possible.
2. Restore the *Content Management Server* with a backup. Note, that this backup must be newer than the backup of the *Master Live Server*.
3. Stop the *Master Live Server*. Restart the *Content Management Server*. Right now, it is possible again to change content, but no new publications are enabled.
4. Recover the *Master Live Server* and the *Replication Live Servers*.

Note that the use of incremental backups is advised in order not to lose any data.

3.9.3 Recovery of a Master Live Server Database

There are two possibilities to recover a corrupted *Master Live Server* database.

Recovery with Master Live Server's backup

1. Stop the *Master Live Server*. The *Replication Live Servers* will still be running. However, no publications will be possible.
2. Restore the backup of the *Master Live Server*.
3. Restart the *Master Live Server*; since the *Replication Live Servers* do have a newer content base than the *Master Live Server*, no replication is possible and the *Replication Live Servers* have to be recovered as well.
4. Recover the *Replication Live Servers* as described in [Section 3.10.4, "Restoring from Replication Live Server Backup" \[56\]](#).

Because a backup for the *Master Live Server* has been restored, all publications executed in the time between creating and restoring the backup are not available

on the Live side. You can use the tool `republish` (see [Section “Republish” \[150\]](#)) to repeat the publications automatically.

Fast Recovery with *Replication Live Server's* backup

Be aware, that it might be necessary to copy the database backup to other *Replication Live Servers*, as well (see step 2).

1. Stop the *Master Live Server*.
2. To select the *Replication Live Server* database from which you want to take the backup, you have to take the current sequence numbers of all *Replication Live Servers*. You get the sequence number by executing the following SQL statement on every *Replication Live Server* database:

```
SELECT * FROM system WHERE property='replicator_remote';
```

The sequence number of some (or all) *Replication Live Servers* will be the same. If not, determine the most often occurring sequence number. In the following, this sequence number is called dedicated sequence number.

Choose one of the *Replication Live Servers* that has the dedicated sequence number.

In the following, this server is called the dedicated *Replication Live Server*. Its data will be used to recover the *Master Live Server* and all *Replication Live Servers* with a different sequence number than the dedicated sequence number.

It is important that the dedicated *Replication Live Server* is in a consistent state. To check this, the table `ReplicatorIdTable` has to be empty. When the table is not empty, the *Replication Live Server* is inconsistent. To see the table content, execute the following statement. If the server is inconsistent, choose another *Replication Live Server* as the dedicated *Replication Live Server*:

```
SELECT * from ReplicatorIdTable;
```

3. Create a backup of the dedicated *Replication Live Server's* database using database tools. Note that the *Replication Live Server* needs not to be stopped.
4. If the passwords on the dedicated *Replication Live Server* have been encrypted via `cm_encryptpasswords`, the Rijndael key of the *Replication Live Server* in the file `WEB-INF/etc/keys/<database>.<dbuser>.rijndael` has to be copied to the respective directory on the *Master Live Server* (`<database name>` and `<dbuser>` have to be adjusted to the appropriate values of the *Master Live Server* database). If the directory does not exist on the *Master Live Server*, you must create it.

The file must also be copied to all *Replication Live Servers* with a different sequence number than the dedicated sequence number.

5. Delete the full *Master Live Server* database schema using database tools.

6. Import the database backup into the database of the *Master Live Server* using database tools.
7. On the *Master Live Server*, fetch the last changelog entry with the following SQL statement and note the values of `<sequenceno>` and `<idtag>`:

```
SELECT * FROM changelog WHERE sequenceno = (SELECT
max(sequenceno) FROM changelog);
```

8. On all *Replication Live Servers* with the same sequence number as the dedicated sequence number, adjust the replicator settings with the following SQL statement (note, that the *Replication Live Servers* do not have to be stopped, but make sure that the *Master Live Server* is still not running). Replace `<sequenceno>` and `<idtag>` with the values obtained above:

```
UPDATE system SET content='<sequenceno>' WHERE prop
erty='replicator_remote';
UPDATE system SET content='<sequenceno>' WHERE prop
erty='replicator_local';
UPDATE system SET content='<idtag>' WHERE property='rep
licator_tag';
```

9. All *Replication Live Servers* with a different sequence number than the dedicated sequence number must be recovered. To recover, execute the steps 5., 6., and 8. (read *Replication Live Server* instead of *Master Live Server*).
10. Restart the *Master Live Server* and the *Replication Live Servers*. Now you can publish content from the *Content Management Server*, and *Replication Live Servers* can replicate the content.

Note, there might be some inconsistencies between *Content Management Server* and *Master Live Server* in respect of some publication events, that did not reach the dedicated *Replication Live Server*. This is the case

- if the dedicated *Replication Live Server* is not the one with the highest sequence number,
- if the dedicated *Replication Live Server* has not been connected to the *Master Live Server* for a longer time, that is that there were publication events that did not reach the *Replication Live Server*,
- if the *Master Live Server* database crash did hinder the replication process between *Master Live Server* and *Replication Live Servers*.

In all these cases, you have to replay all affected publications on the *Content Management Server*. To detect the affected publications, you can use the *Site Manager* to query all publication events in the specific period.

3.10 Administrating Replication Live Servers

The following sub sections explain how to

- install *Replication Live Servers*
- restore a *Replication Live Server* database from a *Replication Live Server* database backup
- restore a *Replication Live Server* database from a *Master Live Server* database backup
- uninstall *Replication Live Servers*

The initial replication of a *Replication Live Server* is not fault-tolerant against connection losses. The first replication has to run without interruption in order to succeed. If this is not possible, the database of the *Replication Live Server* has to be build from a backup or the database of the *Master Live Server* as described in the following sections.



3.10.1 Installing the First Replication Live Server

To install the first *Replication Live Server* proceed as follows:

1. Install the *Replication Live Server* with a *Replication Live Server* license.
2. Configure the content types equal to the types on the *Master Live Server*.
3. Configure the replicator in `properties/corem/replicator.properties` with the IOR URL of the *Master Live Server*:

```
replicator.publicationIorUrl=http://<MasterLiveServerComputerName>:<port>/coremedia/ior
```

4. Because the server will be available to clients immediately, take care to change the passwords of the default users by setting the `cap.server.initialPassword.<USERNAME>` properties as described in [Section 5.1, “Configuration in contentserver.properties” \[227\]](#).
5. Make sure that the property `replicator.tmpDir` points to a directory that has enough free space to hold the blob content of your whole repository that will be replicated.
6. Start the *Replication Live Server*.

The *Replication Live Server* will switch to the run level `administration` and start the replication of the content. When the initial replication is completed, the server will go `online`.



The setup of a *Replication Live Server* must be completed without additional re-starts of the server. If the initial replication fails for any reason, you must empty the database schema and repeat the entire setup from step 3.

If the replication does not complete repeatedly, the slave database has to be build from a backup of the master database as described later in the manual.

3.10.2 Installing further Replication Live Servers

To install further *Replication Live Servers* you can repeat the instructions from [Section 3.10.1, “Installing the First Replication Live Server” \[54\]](#) or you can set up a *Replication Live Server* from the backup of another *Replication Live Server* as described in the following. This is recommended if the repository size is large.

1. Install additional *Replication Live Servers* with *Replication Live Server* licenses.
2. Restore the database with the last database backup of the first *Replication Live Server*.
3. Configure the content types equal to the types of the *Master Live Server*.
4. Configure the replicator in `properties/corem/replicator.properties` with the IOR URL of the *Master Live Server*:

```
replicator.publicationIorUrl=http://<MasterLiveServerComputerName>:<port>/coremedia/ior
```

5. Start the *Replication Live Server*.

The *Replication Live Server* will switch to the run level `administration` and start the replication of the content, processing changes since the backup was created. When at most the number of events specified in the property `replicator.maxAcceptedLag` (see [Section 5.4, “Configuration in replicator.properties” \[247\]](#)) remains unprocessed, the *Replication Live Server* will go online.

3.10.3 Replication Live Servers Backups

The database content of *Replication Live Servers* is exchangeable, that means it is possible to restore the database of one *Replication Live Server* with the database content of another *Replication Live Server*. Complete replication of the *Master Live Server* takes far longer for huge databases than restoring a database with a backup. On average the replication process creates 5-10 content items per second in the repository, depending on the content item size. Therefore, do the following:

- Execute regular database backups of a *Replication Live Server*

- Restore the *Replication Live Server* database with a backup.

Note that *Content Management Server*, *Master Live Server* and *Replication Live Server* databases stand in a fixed temporal order. The state of the *Content Management Server* must be younger than the state of the *Master Live Server*. The state of the *Master Live Server* must be younger than the state of the *Replication Live Server*. Backups must follow this order. Therefore, do the following:

- Create backups in the order *Replication Live Server*, *Master Live Server*, *Content Management Server*.

The following section explains how to restore the database in detail.

3.10.4 Restoring from Replication Live Server Backup

If you must restore the *Master Live Server* database with a backup, you also have to restore all *Replication Live Servers* databases. Proceed as follows:

1. Stop the *Master Live Server*.

The replication on all *Replication Live Servers* stops automatically, but the servers remain online.

2. Restore the *Master Live Server* database with a backup.
3. Restart the *Master Live Server*.

The *Replication Live Servers* now log on again to the *Master Live Server*, discover that the state of the *Master Live Server* database is older than their own state and stop replication, but remain online. To synchronize the databases you can apply the following strategy:

4. Stop the first half set of the *Replication Live Servers* one by one and restore the databases with a *Replication Live Server* backup which is older than the *Master Live Server* backup.
5. Start the *Replication Live Servers*.
6. Continue at step 4 for the second half set of the *Replication Live Servers*.

3.10.5 Restoring from Master Live Server Backup

If you are a skilled administrator, you do not need to create backups from *Replication Live Server*. Instead, you can use a backup from the *Master Live Server*. However, this procedure requires that you execute some SQL statements. To restore a *Replication Live Server* database from a *Master Live Server* backup you have to do the following:



If you have a client connected with your *Replication Live Server* that uses the timestamp of the server (for example, the *CAE Feeder*) you have to find the last "sequenceno" in the "changelog" table of the *Master Live Server* and *Replication Live Server* where both servers were in sync. You also have to check, if the client has already processed this event. Do this, before you replace the *Replication Live Server* database.

Finding the last common sequence number

Finding this sequence number is a bit tricky, because the numbers are different on different *Content Servers*. Therefore, you have to find the last common event in the changelog (see [Section 2.5, "Changelog" \[27\]](#)) tables of both *Content Servers*. To do so, proceed as follows:

1. Find the last event in the changelog table of the *Replication Live Server* that has been processed. The event should not have "code=15" because this indicates a local event:

```
SELECT * FROM changelog WHERE sequenceno = (SELECT max(sequenceno)
FROM changelog WHERE code != 15);
```

2. To find this event in the changelog table of the *Master Live Server* you need the values of the following table columns: Code, I1, I2, I3, B1, B2, S1, S2, S3. Execute the following SQL statement on the *Master Live Server's* database, replace the values in angle brackets with the according values from the *Replication Live Server*, :

```
SELECT sequenceno FROM changelog WHERE code=<ReplicationCode>
AND I1=<ReplicationI1> AND I2=<ReplicationI2> AND I3=<ReplicationI3>
AND B1=<ReplicationB1> AND B2=<ReplicationB2> AND S1=<ReplicationS1>
AND S2=<ReplicationS2> AND S3=<ReplicationS3>
```

This gives you the sequence number for further processing in "Clients and Timestamps" below.

Now, you have to check if the client has already processed the event with this timestamp. Otherwise, you might lose events. In the case of the *CAE Feeder* execute the following SQL script on the *CAE Feeder's* database:

```
SELECT * FROM pcproperties;
```

This returns a result as shown in the following example:

```
com.coremedia.amaro.persistentcache.propertystore.PropertyVerifier
-application caefeeder
com.coremedia.amaro.cae.feeder.IndexVerifier CAEFEEDER:1252067316056
ContentDependencyInvalidator-timestamp 442525:1:866469822
com.coremedia.cap.persistentcache.proactive.content.ContentTrigger.
timestamp.trigger 442366:6:866469822
```

Take the content of the properties with "timestamp" in their name. The first number in the content of these properties is the sequence number of the last event. These numbers must be equal or larger than the common sequence number you have found above. If it is smaller, you have to look for a smaller common sequence number, otherwise you would lose events in the *CAE Feeder*.

Restore the database

1. Stop the *Replication Live Servers*.
2. If the database does not allow you to take online backups ensure that all publications are finished and that the last publication was successful. Freeze the content of the *Master Live Server*, so that no new publications are allowed. If possible, stop the *Master Live Server*.
3. Create a backup of the *Master Live Server* database with the database backup tool.
4. If the passwords on the *Master Live Server* are encrypted with `cm encryptpasswords`, the Rijndael key of the *Master Live Server* in the file `$INSTALLED_DIR/etc/keys/<databasename>.<dbuser>.rijndael` has to be copied to the respective directory on the *Replication Live Servers* (`<databasename>` and `<dbuser>` have to be adjusted to the appropriate values of the *Replication Live Server* database). If the directory does not exist on the *Replication Live Servers*, it has to be created.
5. If stopped, start the *Master Live Server*. You may continue to start publications again.
6. Delete the full *Replication Live Server* database schema, if existent.
7. Restore the database of the *Replication Live Servers* with the backup.
8. On the *Replication Live Server*, fetch the last changelog entry with the following SQL statement and note the values of `<sequenceno>` and `<idtag>`:

```
SELECT * FROM changelog WHERE sequenceno = (SELECT max(sequenceno)
FROM changelog);
```

9. On the *Replication Live Server*, adjust the replicator settings with the following SQL statement. Replace `<sequenceno>` and `<idtag>` with the values obtained above:

```
INSERT INTO system (property, content) VALUES ('replicator_remote',
'<sequenceno>');
INSERT INTO system (property, content) VALUES ('replicator_local',
'<sequenceno>');
INSERT INTO system (property, content) VALUES ('replicator_tag',
'<idtag>');
```

10. Remove the content of the named license tracking table:

```
DELETE FROM CmLicenses
```

11. Set the proper server type.

```
UPDATE system SET content='live' WHERE property=
'repository_server_type' AND content='publication'
```

As a result you must see: Updated 1 rows

12. Restart the *Replication Live Server*.

Clients and Timestamps

The CoreMedia systems uses timestamps to synchronize servers and clients. If you replace the database of a *Replication Live Server* with the database of a *Master Live Server*, the timestamp of the *Replication Live Server* will be replaced by the one of the *Master Live Server*. Every client, that connects with the *Replication Live Server* and that stores and uses the timestamp - such as the *CAE Feeder* - will stop working because the stored timestamp is different from the new timestamp of the *Replication Live Server*.

In order to synchronize client and server again, you have to replace the timestamp stored by the client with the timestamp of the server. To do this for the *CAE Feeder*, proceed as follows (replace the angle brackets with the appropriate value):

1. Find a sequenceno that originates from before the stop of the *Replication Live Server* as described above and that has been processed by the client.
2. Execute the following SQL statement on the *Master Live Server* database:

```
SELECT * FROM changelog WHERE sequenceno=<No from step 1>
```

3. Extract the timestamp from this information. The timestamp has the format: SEQUENCENO:1:IDTAG with the values taken from the row of the user publisher. This timestamp will be called Master Timestamp below.
4. Stop the client.

5. Execute the following SQL statements on the clients database:

```
UPDATE pcproperties SET content='<Master Timestamp>' WHERE
property='com.coremedia.cap.persistentcache.proactive.
content.ContentTrigger.timestamp.trigger';
```

6. Check if the property "ContentDependencyInvalidator-timestamp" exists. If it exists, execute the following statement.

```
UPDATE pcproperties SET content='<Master Timestamp>' WHERE
property='ContentDependencyInvalidator-timestamp';
```

7. Start the *CAE Feeder* again.

Now, the feeder starts working again. Log errors like "Illegal transition for feeder:coremedia:///cap/content/701282:add" are harmless. They only say, that this content has been fed before.

3.10.6 Removing a Replication Live Server

The *Master Live Server* has no information about the connected *Replication Live Servers*. A *Replication Live Server* can simply be stopped and deinstalled.

3.10.7 Analyzing the Replicator State

The replicator component is a *Replication Live Server*. It can be started and stopped while the server itself continues running. It consists of a controller process and a number of stages that process events for the *Master Live Server*. Events are always processed in the order in which they were created.

The replicator depends on the availability of two servers, two databases, and a network connection. The replication fails to make progress if any of these components fails. This section gives some hints for analyzing the replicator state if you suspect that events from the *Master Live Server* are not properly replayed on the *Replication Live Server*.

Checking the Server States

Using

```
cm runlevel -u <user> -p <password>
cm systeminfo -u admin
```

you can verify whether both servers are up and in runlevel online. The system information also tells you whether the initial replication of the *Replication Live Server* has been completed successfully.

Checking the Replicator Configuration

In the file `properties/corem/replicator.properties` look at the IOR URL of the *Master Live Server* in the property `replicator.publicationIorUrl` and at the property `replicator.enable`, which starts and stops the replicator. In case of an error that is not automatically healed for an extended period, it can make sense to set `replicator.autoRestart=false` to ensure that the error condition can be analyzed without continuous restart attempts of the replicator.

Checking Replicator Startup Messages

If in doubt whether and when the replicator was started, check the log for messages of the form

```
[CURRENT_DATE] Info: cap.server.replicator:
Action(name="StartAction", completed=false): running
[CURRENT_DATE] Info: cap.server.replicator:
Replicator: pipeline created
...
[CURRENT_DATE] Info: cap.server.replicator:
Replicator: connected
[CURRENT_DATE] Info: cap.server.replicator:
Action(name="StartAction", completed=true): completed
```

where `CURRENT_DATE` is the date when the replicator was started, typically shortly after the server start. If these messages are repeated over and over again, the connection to the *Master Live Server* might be broken, especially if only the `pipeline created` message is printed, but the replicator never claims to be connected.

Checking Replicator Status Messages

After a successful start, the replicator writes frequent status messages to its log file on the log facility `cap.server.replicator` at log level `info`. A healthy idle replicator looks like this:

```
[CURRENT_DATE] Info: cap.server.replicator: EventStatistics:
Replicator(enabled=true, initialized=false, state="running",
pipelineUp=true, connectionUp=true, alreadySwitched=true,
logEvents=false, autoRestart=true, checkStream=true,
checkTimeout=300)
```

```
[CURRENT_DATE] Info: cap.server.replicator: EventStatistics:
IncomingCounter(lastSequenceNumber=SEQ_NO,
lastStampedNumber=SEQ_NO, count=EVENT_COUNT,
lastEventArrived=LAST_EVENT_DATE,
startedAt=REPLICATOR_START_DATE)
```

```
[CURRENT_DATE] Info: cap.server.replicator: EventStatistics:
CompletedCounter(lastSequenceNumber=SEQ_NO,
lastStampedNumber=SEQ_NO, count=EVENT_COUNT,
lastEventArrived=LAST_EVENT_DATE,
startedAt=REPLICATOR_START_DATE)
```

where `REPLICATOR_START_DATE` is a date very close to the start of the *Replication Live Server* or the last start of the replicator, if the replicator has been restarted since the server start. `LAST_EVENT_DATE` should be a date very close to the last successful publication or the last replicator start, whichever comes later. `SEQ_NO` should be the sequence number of the last event received or 0, if no content has been published since the replicator start.

If messages of this type do not appear once per minute, check the log configuration and use the above mentioned command to check the server state. Make sure the replicator is enabled in the file `replicator.properties`.

The three entries in the log have the following meaning:

The first line tells you to which extent the replicator is up. It also provides some basic configuration information:

- `enabled: true`, if the replicator is enabled in the configuration file `replicator.properties`;
- `initialized: true`, if the initial replication ever completed successfully during a previous or the current run of the server; if the current run performed the initialization, it is necessary that the replicator also caught up with the continuous event stream of the *Master Live Server*;
- `state: running`, if the replicator pipeline is up, `not started`, if the replicator was never started during the current run of the server, `stopped`, if the replicator pipeline has been completely stopped, and `failed` in the rare case that the replicator pipeline controller itself died;
- `pipelineUp: true`, if the replication pipeline is ready to process events; this does not imply that events are actually being retrieved or processed, just that the infrastructure is available;
- `connectionUp: true`, if the connection to the *Master Live Server* has been established successfully;
- `alreadySwitched: true`, if the replicator has at least once caught up with the live event stream from the *Master Live Server*, switching to runlevel online as a result; it is sufficient if it came to within `replicator.maxAcceptedLag` events and dropped back later on;
- `logEvents: true`, if individual events are logged as they propagate through the replicator pipeline;
- `autoRestart: true`, if the replicator restarts automatically, if the event stream from the *Master Live Server* is broken;
- `checkStream: true`, if the replicator checks regularly whether the event stream from the *Master Live Server* is still intact;
- `checkTimeout`: the interval between two checks of the event stream from the *Master Live Server*.

The second line reports on the incoming events from the *Master Live Server*. When a publication is performed, the reported values should quickly rise to the last sequence number reported at the *Master Live Server* by `cm events`.

- `lastSequenceNumber`: the sequence number of the last *Master Live Server* event that arrived at the replicator; this value will stay 0 until the first event is received after a restart and while the initial replication is performed.
- `lastStampedNumber`: the sequence number of the last stamp event from the *Master Live Server*; such an event indicates the end of a publication;
- `count`: the total number of event that arrived since the replicator was started;
- `lastEvenArrived`: the date of the last arrival of an event from the *Master Live Server*;
- `startedAt`: the start date of the replicator.

The third line reports the complete processing of events by the replicator pipeline. The reported properties are identical to the properties reported by the incoming event counter. Normally, the values reported here should lag only slightly behind the properties for the incoming events or should match them exactly. However, there are legitimate reasons for differences:

- During the initial replication, the incoming events may already come from the live event stream, showing a large positive number, while the completed events are still drawn from the initial synthetic replay of the *Master Live Server* repository, showing 0 as the sequence number.
- During times of very high load and after a long downtime, the incoming event might be ahead of the processed events for an extended period, until the replicator has caught up with the live event stream.

Generally, you should not worry about the health of the replicator as long as the property `count` of the processed events is continually rising, because that indicates that events are still being processed.

Interpreting Special Messages

The replicator outputs quite a lot of log messages in special occasions. The most frequent messages will be discussed.

- `replicator still X events behind, will not yet go online`: The replication was started and has just replicated another complete publication, but the live event stream is still way ahead, so that it is not safe to switch the replicator online. Use `cm runlevel` to force a switch.
- `initial replication complete, will not go online`: A freshly installed replicator has finished its initial replication. Use the opportunity to change the various default passwords before switching the server into online mode using `cm runlevel`.

- possibly disconnected from event stream: The replicator suspects that the *Master Live Server* is no longer feeding events. Depending on the configuration, the replicator may restart itself.
- resource to be replicated already destroyed or version to be replicated already destroyed: While processing an event for a resource or a version, that object is no longer readable. It is assumed that a subsequent destroy event has caused this situation. This message only indicates a temporary inconsistency in the repository that will be healed automatically.
- cannot initialize repository as the repository not empty: Typically indicates that a previous initial replication did not complete. The replicator cannot recover from that failure. Drop the database schema and retry, possibly in times of lower load or with more memory allocated to the server process. If the initial replication fails repeatedly, create a new *Replication Live Server* from a backup of the *Master Live Server* as previously discussed.

3.11 Administrating Multi-Site Publishing

3.11.1 Enabling Multi-Site Publishing

You have to enable multi-site publication before the first start of the *Content Management Server*. To do so, you have to add the following property to the `contentserver.properties` file of your *Content Management Server*.

```
cap.server.multipleLiveServers=true
```

It is not possible to switch from an existing single-site system to a multi-site system by simply setting `cap.server.multipleLiveServers=true`. Please read [Section 3.11.4, “Migrate to Multi-Site Management Extension” \[67\]](#) for a detailed description of the migration process.



3.11.2 Configuring Multi-Site Publishing

Configuration of multi-site publishing means configuration of the publication targets. For each target the *Content Management Server* needs a set of properties which have to be defined in the `WEB-INF/properties/corem/publisher.properties` file. You have to replace `<n>` with a consecutive number for each property set, starting with 1.

It is possible to change the publication target configuration after the first start of the server and even if the server is running.

Property	Description
<code>publisher.target.<n>.user</code>	The user which is used by the publisher to log in to the <i>Master Live Server</i> .
<code>publisher.target.<n>.domain</code>	The domain of the user which is used by the publisher to log in to the <i>Master Live Server</i> .
<code>publisher.target.<n>.password</code>	The password of the user which is used by the publisher to log on the <i>Master Live Server</i> . Make sure, that you change the password of the existing user <code>publisher</code> on the <i>Master Live Server</i> or that you newly create this user on the <i>Master Live Server</i> .
<code>publisher.target.<n>.ior.url</code>	The URL where the publisher can obtain the IOR of the <i>Master Live Server</i> .
<code>publisher.target.<n>.name</code>	The permanent and unique name of the publication target. This name is used for target identification in the APIs and in JMX. Changing this name may lead to

Table 3.4. Properties used to configure Multi-Site Management

Property	Description
	unexpected failures in clients that are not properly stopped, changed, and restarted.
<code>publisher.target.<n>.display.name</code>	The display name is shown to users when no localized information about a publication target is available. Display names, too, should be unique, but they may well change to better illustrate the current uses of a publication target.
<code>publisher.target.<n>.folders</code>	<p>The property folders typically references exactly one top-level folder, either by name or by its numerical id. If more than one site is generated from a single <i>Live Server</i>, multiple top-level folders may be given, separated by commas. For example, the following configuration line would be correct:</p> <pre>publisher.target.2.folders=internet,download,5173</pre> <p>It specifies three folders that are mapped to a single target, one of which is the folder with ID 5173. Listing folders numerically can be helpful when a folder must be renamed, but should not leave its publication target. Once you have assigned a folder to a publication target, it must not be reassigned to another target. Doing so would result in inconsistencies between <i>Content Management Server</i> and <i>Master Live Server</i>.</p>

Once you have assigned a folder to a publication target, it must not be reassigned to another target. Doing so would result in inconsistencies between *Content Management Server* and *Master Live Server*.

When you configure base folders by name, a top-level folder of that name may not be renamed and no other top-level folder may be renamed to assume that name. This stops users from reassigning top-level folder accidentally. When you want to rename top-level folders that belong to a publication target, configure them using their numeric id in the file `publisher.properties`.



3.11.3 Adding Publication Targets

Publication targets can be added dynamically as needed. Simply install a new *Master Live Server* and add a new group of properties to the `publisher.properties` file of the *Content Management Server*. The *Content Management Server* will reload the property file automatically within a few seconds and schedule subsequent publications according to the updated information.

Take special care if rules for live groups are defined on the root folder. Such rules are published to all *Master Live Servers* upon creation. When adding a new *Master Live Server*, rules on the root folder are missing in that server at first. You can transfer these rules to the new server, if you log in to the *Content Editor* as user admin, open the user manager window and select **Files|Synchronize live rules**.

3.11.4 Migrate to Multi-Site Management Extension

The wish to perform a migration project from single-site to multi-site mode might arise when a *CoreMedia CMS* system is extended to host an intranet application besides an existing Internet web server. It is also possible that multiple sites that were previously hosted on different servers are about to be unified on a single system.

In multi-site mode, all content that belongs to a publication target should reside below a single top-level folder. Therefore, the main target of the migration is to modify the repository structure to reflect this requirement.

Before the migration, the repository might look like this:

```
/
+--Articles
+--Home
+--Inbox
+--News
+--Pictures
```

Afterwards one more folder has been introduced that hosts most of the repository's content.

```
/
+--Home
+--Live
  +--Articles
  +--Inbox
  +--News
  +--Pictures
```

After determining the new repository structure, the main step of the migration is handled by the `cm multisiteconverter` tool, which updates the database. Still, care has to be taken, because the *Content Management Server* has to be down while doing the conversion. For typical *CoreMedia CMS* installations a considerable number of servers, CAEs, and importers is deployed and an extended downtime that is noticeable to the content consumers is not acceptable. A migration path will be shown, that keeps the visible downtime low.

There will be certain aspects of the process described in the following sections that are not directly applicable to your installation. It is therefore important to spend enough time on finding a process that meets your needs, before starting the actual migration project.

For a start, assume that you are migrating to the *CoreMedia Multisite Management* without changing migrating to a different release at the same time. If you want to do so to avoid multiple downtimes, see [Section “Updating Simultaneously” \[71\]](#). Furthermore, assume that the content that is currently stored in the existing *Content Management Server* should go to a single *Master Live Server*, so that you are free to add further *Master Live Servers* later on. If you want to distribute the existing content over multiple *Master Live Servers*, see [Section “Splitting Content to Multiple Targets” \[72\]](#) for details.

Creating a Test Environment

During migration, a lot of configurations and code has to be validated or adapted to ensure that it will conform with the new repository structure. You have to test the updated configuration files and programs in a realistic environment. To that end, you have to set up a new test environment, that is converted to multi-site mode early on during the migration project. In the following you will learn how to set up such a test system. Most changes are quite similar to the actual conversion process that happens at the time of the relaunch.

1. Create two new database users for a new *Content Management Server* and a new *Master Live Server*.
2. Initialize the database users using consistent backups of the existing servers.
3. Install a new *Content Management Server* and a new *Master Live Server* (release *CoreMedia CMS 2005* or later) and configure them to use the new database users. Keep the servers in single-site mode.
4. Start the servers.
5. Update the folder structure of the *Content Management Server*. Publish the changes. (You might want to perform the required actions using an automated script, so that the script can be reused later on during relaunch.)
6. Shutdown the *Content Management Server*.
7. Make sure that the *Content Management Server* is not already in multi-site mode by verifying that the property `cap.server.multipleLiveServers` in `contentserver.properties` is set to `false`. Run `cm multisiteconverter` in the *Content Management Server Tools* installation. This will automatically perform the following actions:
 - Verifying that the *Content Management Server* was previously in single-site mode and that it is not currently running.
 - Updating the database to include correct base folders for all resources.
 - Updating the database to mark the schema as multi-site enabled.
8. Change the property `cap.server.multipleLiveServers` in `contentserver.properties` to `true`.

9. Start the *Content Management Server*.
10. Update the file `publisher.properties` to include the new top-level folder as its single publication target.

You may now install and use additional clients as needed.

When setting up the new repository structure in step 5, the ideal way is to put all resources into a single top-level folder, except the `Home` and `System` folders (`Home` and `System` folders normally contain no published content, see [Section “Migrating the Clients” \[69\]](#)). If you decide to keep multiple top-level folders, there may remain wide links. Wide links are links from one top-level folder tree to another top-level folder tree. Because such links might potentially span multiple publication targets, leading to local dead links, they are strongly discouraged and are reported as an error during publication. Therefore, you might want to execute a query for wide links in the *CoreMedia Editor* at this point. You might have to rethink your repository structure if wide links cannot be easily cleaned up.

Migrating the Clients

In this chapter it will be discussed if the various repository clients might require an update of their configuration or code. The new configurations or implementations are tested against a test version of the multi-site repository.

Migrating Importers

Importer configurations have to take into account that the target folder for imported resources will change due to the newly inserted top-level folder.

Migrating the User Manager

Normally, the home folders of users are not published, so they may stay in place, not moving them into the new top-level folder. In the case that the home folders should be moved, the user information retrieved from LDAP servers must be updated to reflect the new repository layout. The preferred way of doing this is to use a modified `UserProvider` that handles the change. Users from the built-in user management reference the home folder by ID and as such are more robust with respect to repository restructurings.

Migrating Other Clients

Other clients or workflows, too, might make use of well-known resources for retrieving configuration information or for storing temporary data.

Migrating the Content Servers

When you have ported all clients to the new repository layout in the test environment, the migration must also be performed in the content management environment.

The migration of the *Content Servers* is a critical operation that must be scheduled and supervised most carefully.



Perform a dry run of the following steps firstly using dedicated test servers to gain knowledge about the delays involved in the various steps and to ensure that all steps work correctly.

You have to pay special attention to back up procedures before and during the migration.

1. Make sure that there is an up-to-date backup of the servers and database schemas that will be modified.
2. Make sure that the *Content Management Server* is not already in multi-site mode by verifying that the property `cap.server.multipleLiveServers` in `contentserver.properties` is set to `false`.
3. Using the information gained during the test phase, install new instances of all clients that need updating. Do not yet start the new clients, but make sure that they are correctly prepared for start-up and connection to the main servers.
4. Stop all editorial work.
5. Shutdown or reconfigure the watchdogs as applicable, because many processes are going to be down soon.
6. Stop all writing clients, including, but not limited to, importers and scripts. (Make sure to reschedule all periodic tasks that would happen in the next hours.)
7. Shut down the *Workflow Server*.
8. Shut down the preview *CAE* and all other reading clients that access the *Content Management Server*.
9. Make sure that all *Replication Live Servers* have caught up with the *Master Live Server*. You may use the tool `cm events` on the *Replication Live Servers* for this purpose.
10. Stop all clients that are attached to the *Live Servers*, including, but not limited to, *CAEs*. At this point of time, your live site is down.
11. Update the folder structure of the *Content Management Server*. Publish the changes. (This is typically done using the same script that was used to set up the test environment.)

12. Make sure that all *Replication Live Servers* have caught up with the *Master Live Server*, replicating the change of the repository structure.
13. Start the new *CAE* instances that were previously installed in step 2. At this point of time, your live site is up.
14. Because the *CAEs* are starting up with empty caches, expect high load on the servers at this point of time. Wait until the load has returned to normal levels.
15. Start new instances of other reading clients on the *Replication Live Servers*.
16. Shutdown the *Content Management Server*.
17. Run `cm multisiteconverter` in the directory where the *Content Management Server Tools* are installed and on the host where the *Content Management Server* is supposed to be running. The runtime may vary depending on the actual size of the repository and the performance of the database, but 500 resources per second were achieved in a large example migration.
18. Change the property `cap.server.multipleLiveServers` in `contentserver.properties` to `true`.
19. Start the *Content Management Server*.
20. Update the file `publisher.properties` to include the new top-level folder as its single publication target. To this end, you will have to comment out existing configuration parameters and add new configuration parameters.
21. Start the workflow server.
22. Start the new instances of clients at the *Content Management Server* side. Make sure to keep track of the correct start-up order, if that is required.
23. Start or reconfigure the watchdogs as applicable.
24. Enable periodic tasks, possibly rescheduling next runs as needed based on the actual downtime.
25. Resume editorial work.

Once the migration of the primary publication target is done, set up additional publication targets as needed.

Updating Simultaneously

It is possible to perform an update of the CoreMedia system at the same time when the multi-site mode is enabled. For example, you may want to upgrade from *CoreMedia SCI 4.2* to the current release of CoreMedia CMS at the same time as making the switch to Multi-Site Management. In this case, the new servers and clients to be installed are always using the new release, but they may have to be started in single-site mode once before making the switch.

When installing the new *Content Management Server*, make sure to install a single-site server. This reflects the fact that the database is still in single-site format and

will be changed during the actual migration process. Immediately after step 11 of [Section “Migrating the Content Servers” \[70\]](#), you have to stop all *Live Servers* and start preinstalled instances of *Live Servers* using the new release. This way, another CAE downtime can be avoided. After this point, always use the client and server start scripts of the newly installed servers.

Immediately after step 15 of [Section “Migrating the Content Servers” \[70\]](#), you have to switch to the new installation of the *Content Management Server* for the rest of the procedure. That is, when the instructions require you to start a program, you have to start the newly installed program of the target release. The new *Content Management Server* must initially be configured for single-site mode. Depending on the source and target releases, additional procedures might be mandatory at this point.

If you are doing an upgrade from *CoreMedia SCI 4.2* or earlier to *CoreMedia CMS 2005* or later, you have to take an additional action after step 15. You must start the new *Content Management Server* once and shut it down immediately after it has come up successfully. Only after a *Content Management Server* of *CoreMedia CMS 2005* or later has run at least once, you may proceed with step 16.

Splitting Content to Multiple Targets

The previous migration steps assumed that your existing content is supposed to be part of one publication target and that the migration to the multi-site mode was prompted by the need to add entirely new publication targets. In some cases, you might want to split the existing content among multiple publication targets.

The overall procedure remains the same, but some additional steps become necessary.

First of all, you have to create more than one top-level folder, but still only one folder per site that you will host, distributing the existing content as you see fit. Having assigned the content to top-level folders, you should run a query for wide links in order to detect misplaced resources or conceptual flaws in the design. After you have migrated to multi-site mode, it will be much more expensive to correct such problems.

After the CAEs are up again as per step 12 of [Section “Migrating the Content Servers” \[70\]](#), perform the following additional actions.

1. Install new *Master Live Servers* to bring the overall number of *Master Live Servers* to the desired number of publication targets.
2. Copy, on the database level, the *Master Live Server* database for every new server. Typically, you create a backup and restore it multiple times, now.
3. Perform the same action for *Replication Live Servers*, if needed.

4. Reconfigure the CAEs to use those *Live Servers* that are applicable to the site they are building. At this time, you may switch one CAE at a time, effectively reducing the impact on the availability of your site. If a load balancer is distributing requests, reconfigure that as needed during the restarts.

You can now proceed with the migration as before, making sure to configure all publication targets in the file `publisher.properties` in step 18.

As all *Master Live Servers* were initialized as identical copies of the old *Master Live Server*, they contain content that is not applicable for their publication target. You can destroy that content at any time after the migration is complete.

1. Make sure that there is a current backup of the servers and database schemas that will be modified.
2. Make sure that all CAEs and other clients of the *Master Live Server* and its associated *Live Servers* are only using those folders that are published on the *Replication Live Servers* and not those folders that are to be deleted.
3. In the `publisher.properties` of the *Content Management Server*, disable the affected publication target by assigning no folder to the publication target.
4. On the *Master Live Server*, run `'cm multisitecleanup -u publisher -p <PublisherPassword> <FolderNames>'`, including the name of every top-level folder that you want to destroy on the command line. Be careful what content you destroy on which *Master Live Server*. The content that is applicable for the publication target must not be deleted, obviously. The cleanup tool will check whether some of the resources that are to be destroyed are still referenced from the outside. In this case, a destruction is not possible. The tool only destroys content that do not cause dead links. You may have to republish corrected versions of externally referring content item and repeat the cleanup, if the destruction fails.
5. Reenable the publication target in the *Content Management Server* configuration.
6. If step 4 has left part of the resources undestroyed, you can now fix the offending links by changing the content item on the *Content Management Server*, creating two new versions of the offending content items, and publishing both content items in succession. Because the *Master Live Server* stores only two versions of any content item, all old versions on the *Master Live Server* will disappear. Afterwards, you can go back to step 2 and try to clean up the remaining resources.

This cleanup step should be performed during times of low load, but causes no restarts or cache refills.

Content items in the Recycle Bin

The actions described in this section might be required if there are content items in the content repository that were moved into the recycle bin using release 4.1

or earlier. If the previous folder of such content items was destroyed in the mean time, the automatic update procedure will not be able to determine a proper base folder for them. They will be treated as though they were located in the root folder: content items have the root folder as the base folder. This means that such content items can only be restored into the root folder when removing them from the recycle bin. If this is not acceptable for you, then you must restore all resources from trash into a temporary folder that is placed in the proper base folder during migration. After the migration the temporary folder and the recovered content items can be move to the recycle bin again. For the recovery of resources from the recycle bin, you may use scripted queries or editor queries at your discretion.

High Availability During the Migration

Modifying an existing CoreMedia installation, as described in the previous sections, results in a period during which the system is unavailable. The duration of that period is expected to be small, typically a few minutes, if the migration is properly planned and tested. After the migration the server and CAE caches are refilled, which can be a considerable time, too, depending on the actual load profile. If the overall availability expectations are not fulfilled using this scheme, it is suggested to use multiple *Replication Live Servers* of which only one *Replication Live Server* is allowed to replicate the repository structure change at a time, whereas the other *Replication Live Servers* must have stopped replication. Only the CAE attached to the *Live Server* that is currently replicating have to be switched off and replaced by new CAE that can cope with the new repository structure. This way, most of the CAEs will be up at any given time. This is complex and not normally necessary, though. It also requires much more project-specific planning, so that a detailed description is omitted here. In many cases the downtime during steps 9 to 12 of [Section “Migrating the Content Servers” \[70\]](#) can be bridged by setting up a web server delivering a set of static pregenerated pages, if that is needed.

3.12 Truncate the ChangeLog

As described in [Section 2.5, “Changelog” \[27\]](#), all events are written to the `ChangeLog` table of the database. The table is never truncated by default in order to enable event replay right from the beginning. Nevertheless, you might want to reduce the size of the evergrowing table on your own. In one go, you can also truncate the `linkchangelog` table. That is, because the `sequenceno` in this table is a foreign key to the `sequenceno` in the `changelog` table and therefore useless if you delete the respective row in the `changelog` table.

If you delete events from the `changelog`, listeners which try to retrieve these events will fail. In particular, *Replication Live Server* that have not caught up with the *Master Live Server* will become unusable.



Prerequisites

If you want to truncate the `ChangeLog` table the following prerequisites should be fulfilled:

- Little system load for example in the evening.
- All clients which use the `ChangeLog` are up to date.
- New backup of the system has been made

Truncate the `ChangeLog`

When you truncate the `ChangeLog`, you should not delete all entries. You may leave 100000 entries which is a good value proven by practice. Proceed as follows:

1. Get the maximum sequence number from the database using the following SQL statement:

```
select max(sequenceno) from changelog
```

2. Delete all but the last 100000 entries from the table. Replace `<LimitSequenceNo>` with the maximum sequence number minus 100000:

```
delete from changelog where sequenceno < <LimitSequenceNo>
```

3. Delete also all but the last 100000 entries from the `linkchangelog` table.

```
delete from linkchangelog where sequenceno < <LimitSequenceNo>
```

All but the last 100000 entries are deleted from the `changelog` and `linkchangelog` table.

3.13 LDAP Integration

CoreMedia CMS is able to import users and groups from LDAP servers which makes it unnecessary for administrators to manage them in *CoreMedia CMS*. There is an out-of-the-box integration for Active Directory and a customizable support for any LDAP schema. Several LDAP servers can be integrated.

CoreMedia CMS does not write on an LDAP server. You cannot change LDAP memberships with the *CM User Manager* and you cannot create memberships between groups and users from different LDAP servers. However, rules, even for LDAP groups, remain in the *CoreMedia CMS* repository. As they refer to *CoreMedia* resources and resource types, they are repository specific.

CoreMedia CMS clients don't communicate directly with an LDAP server, but get users and groups from the *Content Server*. The *Content Server* accesses users and groups from instances of the interface `com.coremedia.ldap.UserProvider2`. The following subsection explains how `UserProviders` are configured.

3.13.1 User Authentication

CoreMedia CMS supports built-in users and users from external sources like LDAP servers. The content server authenticates built-in users, whereas authentication of LDAP users is delegated to the LDAP server. Authentication is now based on JAAS. Different JAAS login modules authenticate users from different sources. Login modules are Java classes that have to implement the interface `javax.security.auth.spi.LoginModule` (see <http://java.sun.com/products/jaas/>). *CoreMedia CMS* provides default login modules for built-in user and LDAP user authentication:

→ CapLoginModule

The class `hox.corem.server.CapLoginModule` authenticates built-in users. Built-in users are system users created at *Content Server* initialization time and those created later by an administrator with the *CM User Manager*. This module is mandatory, because some system services are run by built-in system users.

→ LdapLoginModule

The class `hox.corem.login.LdapLoginModule` authenticates users from LDAP servers.

You can implement your own login module classes to authenticate users from other origins. Login modules are configured in the `jaas.conf` file.

LoginModule Configuration in `jaas.conf`

The `properties/corem/jaas.conf` file contains the following default configuration for login modules:

Example 3.6. The `jaas.conf` file

```
JaasCap {
    hox.corem.server.CapLoginModule sufficient

    /* System builtin users are not allowed to use the
    editor service */
    predicate.1.class="hox.corem.login.NameLoginPredicate"
    predicate.1.args="negative=true,editor.regex=
    (serverdump|publisher|watchdog|workflow
    |webserver|importer|feeder)"

    /* only specific system user is allowed for the respective
    service */
    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"

    /* if not forbidden by other rules, other services are
    accessible for all users */
    predicate.3.class="hox.corem.login.NameLoginPredicate"
    predicate.3.args="editor.regex=.*,debug.regex=.*,
    filesystem.regex=.*,importer.regex=.*,system.regex=.*,
    dotnetappbridge.regex=.*"
;
/*
hox.corem.login.LdapLoginModule sufficient
host="@ldap.host@" port="@ldap.port@"
domain="@ldap.domain@";

    predicate.1.class="hox.corem.login.NameLoginPredicate"
    predicate.1.args="editor.regex=.*,debug.regex=.*,
    filesystem.regex=.*,importer.regex=.*,system.regex=.*,
    dotnetappbridge.regex=.*"
;
*/
};
```

You have to replace the placeholders `@ldap.host@`, `@ldap.port@` and `@ldap.domain@` for the `LdapLoginModule` with your actual settings.



The syntax conforms to the default configuration syntax in JAAS:

```
Application {
    ModuleClass Flag ModuleOptions;
    ModuleClass Flag ModuleOptions;
    ModuleClass Flag ModuleOptions;
};
```

Example 3.7. JAAS syntax

The syntax is defined in detail in the Javadoc for the Java class `javax.security.auth.login.Configuration`. The application name in `jaas.conf` is `JaasCap`. This value is fix and must not be changed. The `ModuleClass` values in `jaas.conf` can be one of:

- ➔ `hox.corem.server.CapLoginModule`,
- ➔ `hox.corem.login.LdapLoginModule`

→ or another user-defined `LoginModule`.

The value of `Flag` is always set to `sufficient`. The `ModuleOptions` is a space separated list of login module-related arguments. For each domain especially sub domains you have to configure a dedicated `LdapLoginModule` block containing the corresponding domain in the 'domain' attribute of the key 'host'.

The following table contains the module options of `CapLoginModule`:

CapLoginModule Options	Description
<code>predicate.<n>.class</code>	Java class name of a login predicate
<code>predicate.<n>.args</code>	Arguments to the login predicate. Use space value “ ” for no arguments.

Table 3.5. Options of `CapLoginModule`

The two login predicate module options are optional. `<n>` is an integer value. If a login predicate is configured, both options must be set. It is not allowed to omit the second option in the table, for instance. Login predicates are described in [Section “License Management and Login Predicates” \[79\]](#). The following table contains the module options for `LdapLoginModule`.

LdapLoginModule Options	Description
<code>predicate.<n>.class</code>	Java class name of a login predicate
<code>predicate.<n>.args</code>	Arguments for the login predicate. Use space value “ ” for no arguments.
<code>host</code>	LDAP server host name
<code>port</code>	LDAP server port number. If you want to use LDAP over SSL switch to 636, which is the default port for SSL connection.
<code>domain</code>	Domain to serve
<code>protocol</code>	The protocol to use. Change to "ssl" if you want to use LDAP over SSL.

Table 3.6. Options of the `LdapLoginModule`

The meaning of the first two predicate options for the `LdapLoginModule` is the same as in the `CapLoginModule`. The last three options are mandatory.

License Management and Login Predicates

CoreMedia CMS license management is based on named licenses. With named licenses only a fixed and determined number of users can log on to the `contentserver`

for a certain service. The following table lists the services and the associated types of CoreMedia applications.

Table 3.7. CoreMedia services

Service	Applications
debug	Applications used for debugging like <code>cm dump</code>
editor	Editor applications like <i>Site Manager</i>
filesystem	WebDAV
importer	<i>CoreMedia Importer</i>
publisher	Publisher in <i>contentserver</i>
replicator	Replicator in <i>CoreMedia Replication Live Server</i>
system	System applications like <code>cm documentcollector</code>
workflow	<i>CoreMedia Workflow Server</i>
feeder	<i>Content Feeder</i>

When a user, no matter if built-in or LDAP user, logs on to the *Content Server*, he needs a named license for the desired service. The current named licenses in use are stored in a database. If a user logs on for the first time, the user name and the requested service are added to the database. If the maximum number of used named licenses for a service type is reached no further logins for users without named licenses are allowed. The authentication algorithm validates if the users listed in the license table still exist. If not, all used licenses for the non existing users are released and the user may log on. Otherwise, the login fails.

Example:

Assume that you have two licenses for the editor service. When user A is logged in on the *Site Manager*, user A consumes the first license. When a second user B logs on to the *Site Manager* he consumes the second and last free named license. Now no other user than A and B can log on to the editor, even if A or B or both log out again. The two named licenses for the *Site Manager* are reserved for user A and B, until A or B is deleted.

Note: An administrator can manually query and remove used licenses from the database table. See the server tool `cm usedlicenses` in [Section “Usedlicenses” \[141\]](#).



LoginPredicates

The login modules described in the previous section implement user authentication. So far a user may or may not log on to the *contentserver* whatever CoreMedia service he wants to use. The `CapLoginModule` and the `IdapLoginModule` can be con-

figured to restrict user access to certain CoreMedia services. To do this, the two login modules use login predicates. A login predicate is a Java class which implements the interface `LoginPredicate` with the `allowLogin()` method. This method is invoked with a user and a CoreMedia service and returns a `boolean` value. A login module may use several login predicates. Login predicates are evaluated before user authentication in the login modules is actually performed and named licenses are consumed. The evaluation order for login predicates in a login module is as follows:

If no login predicate is specified in a login module, user authentication is performed against the login module.

If multiple login predicates are specified in a login module, they are evaluated in order of their index number `<n>`. If a login predicate for a user and service returns

- *false*, then the user fails to log on immediately.
- *true*, then user authentication is performed against the login module, unless a following login predicate fails.
- *null*, then the result depends on the result of the other login predicates.

The user fails to log on if all login predicates return null.

The login predicate Java classes are:

- `hox.corem.login.NameLoginPredicate`
- `hox.corem.login.AttributeLoginPredicate`
- `hox.corem.login.JndiNameLoginPredicate`
- `hox.corem.login.TrueLoginPredicate`

In the following these classes are described in detail.

NameLoginPredicate

This predicate returns true if the users name or the names of his groups match a regular expression, or null, if not:

NameLoginPredicate Options	Description
<i>negative</i>	Either 'true' or 'false' (default), if the Boolean value returned by the predicate is not null, then it is reversed. A true value is reversed to false and a false value is reversed to true.

Table 3.8. NameLogin-Predicate options

NameLoginPredicate Options	Description
<code>depth</code>	<p>The NameLoginPredicate is not restricted to the user's name but may also consider the names of the groups, he is member of. The integer value specifies the depth of group nesting (defaults to 0). '-1' means the nesting is unrestricted. The values are logically ordered. If one of the values matches the regular expression, the predicate returns true.</p> <p>For example, with depth=0 only the name of the user is used. With depth=1 the name of the user and the names of his direct groups are matched against the regular expression.</p>
<code><service>.regex</code>	<p><code>java.util.regex.Pattern</code> regular expression value where <code><service></code> specifies a service name the pattern is mapped to. Only users matching the given service name and the given regular expression are allowed to log on. <code><service></code> must be one of "debug", "editor", "filesystem", "importer", "publisher", "replicator", "system", "webserver", "workflow", or "feeder".</p>
<code>regex</code>	<p>A <code>java.util.regex.Pattern</code> regular expression value which is mapped to all services, that means <code>regex=some regular expression</code> is an abbreviation for repeating the given expression for all possible services.</p>

AttributeLoginPredicate

This predicate returns true if the users LDAP attributes or the attributes of his groups match a regular expression or null if not.

AttributeLoginPredicate Options	Description
<code>negative</code>	Same as for NameLoginPredicate
<code>depth</code>	<p>The AttributeLoginPredicate is not restricted to the user's attributes but may also consider the attributes of the groups, he is member of. The integer value specifies the depth of group nesting (defaults to 0). '-1' means the nesting is unrestricted. The attributes are the strings returned by the method call <code>hox.com.usermanager.User#getAttributeKeys()</code>. For the LDAP server, these are the values configured with the properties <code>com.coremedia.ldap.user.customattrs</code> for a user and <code>com.coremedia.ldap.group.customattrs</code></p>

Table 3.9. AttributeLoginPredicate options

AttributeLoginPredicate Options	Description
	<p>for a group in the <code>jndi-ad.properties</code> file. The values are logically or-ed. If one of the values matches the regular expression, the predicate returns true.</p> <p>For example, with <code>depth=0</code> only the LDAP attributes of the user are used. With <code>depth=1</code> the LDAP attributes of the user and the LDAP attributes of his direct groups are matched against the regular expression.</p>
<code><service>.regex</code>	Same as for NameLoginPredicate
<code>regex</code>	Same as for NameLoginPredicate

JndiNameLoginPredicate

This predicate returns true if the users JNDI name or the JNDI names of his groups match a regular expression or null if not. The JNDI name is the LDAP distinguished name, that is the reference to the LDAP user entry:

JndiNameLoginPredicate Options	Description
<code>negative</code>	Same as for NameLoginPredicate
<code>depth</code>	<p>The <code>JndiNameLoginPredicate</code> is not restricted to the user's JNDI name but may also consider the JNDI names of the groups, he is member of. The integer value specifies the depth of group nesting (defaults to 0). '-1' means the nesting is unrestricted. The values are logically or-ed. If one of the values matches the regular expression, the predicate returns true.</p> <p>For example, with <code>depth=0</code> only the JNDI name of the user is used. With <code>depth=1</code> the JNDI names of the user and the JNDI names of his direct groups are matched against the regular expression.</p>
<code><service>.regex</code>	Same as for NameLoginPredicate
<code>regex</code>	Same as for NameLoginPredicate

Table 3.10. *JndiNameLoginPredicate options*

TrueLoginPredicate

This predicate always returns true. It has no options but you must configure the `predicate.<n>.args` property with a space character; because JAAS complains if the space is missing.

Example:

A simplified `jaas.conf` file might use `NameLoginPredicate` and `TrueLoginPredicate` with `CapLoginModule`:

```

hox.corem.server.CapLoginModule sufficient

/* System builtin users are not allowed to use the editor
service */
predicate.1.class="hox.corem.login.NameLoginPredicate"
predicate.1.args="negative=true,editor.regex=
(serverdump|publisher|watchdog|workflow|
webserver|importer) "

/* All other users may login */
predicate.2.class="hox.corem.login.TrueLoginPredicate"
predicate.2.args=" "

```

Example 3.8. jaas.conf example

The result of matching the regular expression `(serverdump|publisher|watchdog|workflow|webserver|importer)` is negated for users requesting the editor service. This means that the built-in users with the names `serverdump`, `publisher`, `watchdog`, `workflow`, `webserver` and `importer` are not allowed to use the editor. `TrueLoginPredicate` allows other users to start the editor and all users to use non-editor services.

Evaluation of login predicates is logged on the `contentserver` in the debug log level. If you want to see login predicate results switch the log level from `info` (default value) to `debug` and look for the word 'predicate' in the `contentserver` log after a user has tried to log on. You do not need to restart the `contentserver` when changing the log level.

3.13.2 Configuration of UserProviders

The `UserProvider` instances are configured in the `properties/corem/contentserver.properties` file. Here is an example configuration for an Active Directory Server:

```

cap.server.ldap.1.class=
com.coremedia.ldap.ad.ActiveDirectoryUserProvider
cap.server.ldap.1.properties=properties/corem/jndi-ad.properties

```

Example 3.9. Example for Active Directory Server

The first property configures the `UserProvider` class. The second property refers to a property file holding the provider configuration. Further providers can be added by incrementing the index value in the property keys. The following example shows the configuration of two providers:

```

cap.server.ldap.1.class=
com.coremedia.ldap.ad.ActiveDirectoryUserProvider
cap.server.ldap.1.properties=properties/corem/jndi-ad.properties
cap.server.ldap.2.class=com.mycom.AnotherUserProvider
cap.server.ldap.2.properties=
properties/corem/another_provider.properties

```

Example 3.10. Configuration of two providers

The following sections introduce some predefined user providers classes.

3.13.3 LdapUserProvider

The class `com.coremedia.ldap.LdapUserProvider` is an abstract base class to fetch LDAP entries. It provides all general LDAP-related functionality, especially session handling, searching and caching. It is abstract concerning the membership relation which depends on the particular underlying LDAP schema. This class can not be configured as provider class in `contentserver.properties` but can be subclassed by concrete provider classes. It reads property files like the following example:

```
java.naming.security.principal=CN=Administrator,CN=Users,
DC=example,DC=org
java.naming.security.credentials=admin password
com.coremedia.ldap.host=theLdapServer
com.coremedia.ldap.port=389
```

Example 3.11. Property file

The first two properties specify the account to be used by the *Content Server* for log on to the LDAP Server. The last two properties configure the LDAP host and port number. All properties which start with 'java.naming' are evaluated by the JNDI framework. They are not robust concerning trailing whitespaces, so edit them carefully. The last two properties are mandatory.

```
com.coremedia.ldap.basedns=OU=Accounts,DC=example,DC=org;
OU=Groups,DC=example,DC=org
com.coremedia.ldap.domains=example.org
com.coremedia.ldap.expiration=3600
com.coremedia.ldap.user.customattrs=mail
com.coremedia.ldap.group.customattrs=
```

Example 3.12. Property file

The first property configures a semicolon separated list of base distinguished names (DNs) under which groups and users must be located to be accessible by a *Content Server*.

Due to a bug in the JNDI implementation in Java 8 CoreMedia's Ldap integration cannot correctly cope with referrals. If you use Active Directory with Trust Relationships, you can only use DN's underneath the DC level. For instance, it works fine with `CN=Users,DC=example,DC=org`, but it crashes with `DC=example,DC=org`. The problem manifests in exceptions like



```
2016-03-01 16:24:23 [ERROR] com.coremedia.ldap.LdapUserProvider - Exception while serving
com.coremedia.ldap.ad.ActiveDirectoryUserProvider#getUser: java.util.Vector cannot be cast
to java.lang.String
java.lang.ClassCastException: java.util.Vector cannot be cast to java.lang.String
at com.sun.jndi.ldap.LdapReferralException.getNextReferral(LdapReferralException.java:241)
~[na:1.8.0_65]
at com.sun.jndi.ldap.LdapReferralException.skipReferral(LdapReferralException.java:201) ~[na:1.8.0_65]
at com.coremedia.ldap.impl.LdapConnector.LdapSearch(LdapConnector.java:335)
~[coremedia-ldap-7.1.11-5.jar:7.1.11-5]
```

The second property configures a semicolon separated list of domain names, served by the LDAP server. The third value configures the interval in seconds after which a cached LDAP user in the *Content Server* is updated with modifications made in the LDAP server. Do not set a small value here, as this might lead to performance problems. The last two properties are mandatory and hold a single space separated list of attribute names that are added to and accessible from the `LdapUser` and `LdapGroup` Java objects.

```
com.coremedia.ldap.ou.filter=(objectClass=organizationalUnit)
com.coremedia.ldap.member.filter=
```

Example 3.13. Properties

You may specify two sorts of filters: OU filters and member filters. Using an OU filter allows fine grained configuration of the LDAP nodes to search in, but their usage is not recommended, since it prevents subtree scope search and is thus not performing. The value of the property must follow the syntax described in RFC 2254. You may specify a member filter if needed. A member filter is a class which implements the `com.coremedia.ldap.MemberFilter` interface.

3.13.4 ActiveDirectoryUserProvider

CoreMedia CMS ships with a `com.coremedia.ldap.UserProvider2` class for accessing Microsoft's Active Directory Server: The `com.coremedia.ldap.ad.ActiveDirectoryUserProvider`. For using it you have to configure the following:

1. Tell the *Content Server* to use an Active Directory Server for authentication by adding the following two lines in `WEB-INF/properties/corem/contentserver.properties`. (If you configure multiple UserProviders, take care for the grouping numbers in the property keys.)

```
cap.server.ldap.1.class= \
com.coremedia.ldap.ad.ActiveDirectoryUserProvider
cap.server.ldap.1.properties=properties/corem/jndi-ad.properties
```

2. Adjust the environment specific Active Directory Server properties in `WEB-INF/properties/corem/jndi-ad.properties` as follows:
 - a. Set your Active Directory Servers host and port:

```
com.coremedia.ldap.host=<your-active-directory-servers-host>
com.coremedia.ldap.port=<your-active-directory-servers-port>
```

- b. Set your Administrators domain and password:

```
java.naming.security.principal=CN=Administrator,CN=Users,DC=
\
<your>[,DC=<domain>]*
java.naming.security.credentials=<password>
```

- c. Define the base distinguished names where the `UserProvider` may find users and groups. You can define more than one base DN, separated by semicolons (see also ???).

```
com.coremedia.ldap.basedns=CN=Users,DC=<your>, [DC=<domain>]*
```

3. Activate the `hox.corem.login.LdapLoginModule` within `WEB-INF/properties/corem/jaas.conf`:
 - a. At the end of the file you will find a section, defining the needed login module. Activate it by commenting it out.
 - b. Set the host and port of your Active Directory Server into the corresponding attributes of the login module.
 - c. Set the domain which you chose as domain beneath which your user accounts are stored in step 2.3 above.

Before you may use your Active Directory Accounts within your CoreMedia CMS, you have to define rules for all the given groups your CMS user may be members of. You have to do this as user `admin`. Remember that all CoreMedia system users are not administrated within the Active Directory or any other LDAP server but only from inside of the CoreMedia system itself. Thus, you must not choose any domain when logging into the CoreMedia CMS as user `admin`.

The above description is made for a Windows Server 2008. **Windows Server 2000/2003** are using slightly different LDAP schemas. Basically users and groups are stored in different common names. All user accounts are stored beneath `CN=Accounts,DC=<your>[,DC=<domain>]*` whereas groups are stored beneath `CN=Groups,DC=<your>[,DC=<domain>]*`. In order to connect to an Active Directory Server on such an environment you have to change the following property:

1. Adjust the base dns from step 2.c as follows: `com.coremedia.ldap.basedns=OU=Accounts,DC=<your>, [DC=<domain>]*;OU=Groups,DC=<your>[,DC=<domain>]*`



Common Customizations

By default the `ActiveDirectoryUserProvider` uses the `userPrincipalName` attribute of Active Directory's standard schema as value for a user's name and domain. However, in some projects this is not suitable for whatever reason. This section sketches popular customizations concerning users' names and domains. The approaches assume that all users are unique with respect to the considered attributes. In particular cases this may not be given, and your `UserProvider` might need additional refinements.

Deviating User Domains

In Active Directory a user's domain is modeled in two ways: It can be derived from the DC components of the Distinguished Name, and it is part of the `userPrincipalName` attribute. Usually the two domains are consistent. If the domains by Distinguished Name and by `userPrincipalName` are ambiguous for some users however, you must decide with which domain you want your users to appear in the CMS.

Domain from `userPrincipalName`

If you want to use the domains from the `userPrincipalName`, you must specify the UserProvider's domains explicitly, because they cannot be derived from the configured basedns:

```
com.coremedia.ldap.domains=example.org;other.domain.org;...
```

Include all user domains that may occur in `userPrincipalName` attributes, and all groups' domains. The latter still correspond to the Distinguished Names and must thus be consistent with the basedns.

For the `LdapLoginModule` you must configure the domain of the `userPrincipalName` in `jaas.conf`. If there are multiple user domains, you need an `LdapLoginModule` for each domain.

Domain from Distinguished Name

If you want to use the domains from the distinguished names, you must handle the `userPrincipalName` attribute appropriately, so that the domain part is ignored. Override the following methods of the `ActiveDirectoryUserProvider`:

```
protected String userDomainToAttrValue(String name, String domain, Properties props, int mode) {
    return null;
}

protected String attrValueToUserDomain(Object domain, String dn, Attributes attrs, Properties props) {
    return dnToDomain(dn);
}

protected String userNameToAttrValue(String name, String domain, Properties props, int mode) {
    return ldapSearchTerm(name, mode) + "@" + domain;
}
```

Alternative User Name Attributes

E-mail

The standard AD schema features the `mail` attribute ("E-mail" in the AD UI), which has the same format "name@domain" as `userPrincipalName`. So you can easily switch the attribute. Moreover, you have to add the `mail` attribute to `user.attrs` in order to be fetched at all, and handle the ambiguous domains as described in [section "Domain from `userPrincipalName`" \[88\]](#). In `jndi-ad.properties` the change looks like this:

```
com.coremedia.ldap.domains=example.org;other.domain.org;...
com.coremedia.ldap.user.attrs=... mail
com.coremedia.ldap.user.filter=(o(objectClass=user)(mail=*))
```

```
com.coremedia.ldap.user.nameattr=mail
com.coremedia.ldap.user.domainattr=mail
```

sAMAccountName

Unlike the `userPrincipalName` the `sAMAccountName` attribute consists only of the actual name. So you must override the methods of the `ActiveDirectoryUserProvider` that deal with conversion of name and domain values:

```
protected String userNameToAttrValue(String name, String domain, Properties props, int mode) {
    return ldapSearchTerm(name, mode);
}

protected String attrValueToUserName(Object name, String dn, Attributes attrs, Properties props) {
    return (String)name;
}

protected String userDomainToAttrValue(String name, String domain, Properties props, int mode) {
    return null;
}

protected String attrValueToUserDomain(Object domain, String dn, Attributes attrs, Properties props) {
    return dnToDomain(dn);
}
```

Next, exchange the name attribute in the `jndi-ad.properties`:

```
com.coremedia.ldap.user.filter=(objectClass=user)(sAMAccountName=*)
com.coremedia.ldap.user.nameattr=sAMAccountName
```

3.13.5 Connecting LDAP Over SSL

If you want to connect the *Content Server* via SSL to the Active Directory, proceed as follows:

1. Import all of your needed certificates for the Content Server using the Java *keytool* tool the default password is "changeit" `keytool -import -file ADCert.der -alias ADCert -keystore $JAVA_HOME/jre/lib/security/cacerts`
2. Change the following properties in the `WEB-INF/properties/corem/jndi-ad.properties` file:

```
java.naming.security.protocol=ssl
com.coremedia.ldap.host=ADServer
com.coremedia.ldap.port=636
```

3. Configure the `hox.corem.login.LdapLoginModule` in the file `WEB-INF/properties/corem/jaas.conf` to use SSL by setting the attributes `port="636"` and `protocol="ssl"`.

Now you are done and the *Content Server* will connect to the Active Directory server via SSL.

3.14 Server Utility Programs

The *CoreMedia CMS* provides a series of server utility programs for information, adjustment and optimization of the server processes. Basically there are four categories of such tools:

Categories of server tools

Information

The informational tools allow you to inspect the current state of the *CoreMedia CMS*. You can dump all kinds of server objects, such as resources, running processes or used licenses. Furthermore, you can query system information about the *CoreMedia CMS* itself (esp. the version), which becomes important whenever you submit a support request.

Operation

With the operational tools you control the *CoreMedia CMS*. This includes cleaning up the repository, uploading new workflow definitions and maintaining the database.

Repository

The repository tools allow you to execute actions normally performed by the editors, like approving and publishing resources or starting workflows. They are emergency tools, needed only in exceptional cases to fix the repository immediately when something went wrong.

General usage

General usage

All server utilities are implemented as `cm` programs as described in the [Operations Basics Manual], that is you run them from the command line with `cm`, `cmw` or `cm64`. Most of the tools open a session and therefore need a user who is specified by three command line options `-u`, `-d` and `-p`.

Furthermore, you can explicitly specify the IOR URL of the content server to connect to. If you don't specify the URL, it is taken from `capclient.properties`.

Parameter	Description
<code>-u <name></code>	The name of the user
<code>-d <domain></code>	The domain of the user (optional, only for LDAP users)
<code>-p <password></code>	The password of the user. This is optional. If you don't specify the password on the command line (because you don't want it to be stored in the shell history, for example) the tool will prompt for it at runtime.

Table 3.11. Common options of server utilities

Parameter	Description
<code>-url <ior url></code>	The IOR URL of the content server (optional)

These options are common for most of the server utilities, and will therefore not be mentioned in detail for each tool. Note that most server utilities require administrative permissions and thus can only be run with users of administrative groups.

If you are not sure how to use a utility, just start it without any options to print out a summary of its usage.

General usage in a Windows 64-bit environment

The server utilities can be started using the `cm64.exe` command in a Windows 64-bit environment with a JVM 64-bit, as described in the [Operations Basics Manual].

3.14.1 Information

This section covers the informational server utilities. All these tools are "read only", so none of them modifies the server or the repository.

Informational tools

- `dump`: Shows detailed data of any CMS object (resources, processes, users, ...)
- `encryptpasswordproperty`: Encrypt a given password for use in property files
- `events`: Shows events at the *Content Server* and *Workflow Server*.
- `ior`: Shows the content of an IOR in a readable format
- `license`: Shows the configured *CoreMedia CMS* license data
- `processorusage`: Show used processors
- `publications`: Shows all running publications
- `repositorystatistics`: Show a simple overview over the content in the *CoreMedia* repository.
- `rules`: Shows all rules declared in this repository
- `sessions`: Shows all open sessions
- `systeminfo`: Shows *CoreMedia CMS* installation details
- `validate-multisite`: Validates set up and content structure for multi-site feature

Dump

The `dump` utility prints out arbitrary objects of the CoreMedia CMS repository. You will probably use it mainly to dump resources, but you can also dump users, processes etc.

Usage of dump

```
usage: cm dump -u <user> [other options] [<id1> <id2> ... |
-t <path1> <path2> ... |
-pn <name1> <name2> ... |
-cn <name1> <name2> ... |
-un <name1> <name2> ... |
-gn <name1> <name2> ...]
If not properly prefixed, IDs are interpreted as numeric
content IDs, content repository paths, user names, group
names or content type names as appropriate.

Available options:

-f,--file-name <file name>      dump the output to the
                                given file (using UTF-8)
-d,--domain                      domain for login
                                (default=<builtin>)
-e,--encoding                   output encoding for dump
                                (default=platform encoding)
-cn,--contenttype-name          names of contenttypes to dump
-gn,--group <group>            names of groups to dump
-p,--password                   password for login
-pn,--processdefinition-name    names of processdefinitions
                                to dump
-t,--paths                      path of contents to dump
-u,--user                       user for login (required)
-un,--user-name <user>        names of users to dump
-url <ior url>                 url to connect to
-v,--verbose                    enables verbose output
```

The options have the following meaning:

Parameter	Description
-f <file name>	Name of the file where to dump the output using UTF-8 encoding. By default, the output is written to stdout.
-e <encoding>	Choose the encoding of the dump output. "ISO-8859-1" creates a dump in ISO-Latin-1. "UTF-8" creates a Unicode dump. The default value is the platform encoding. Consult the API documentation of <code>java.nio.charset.Charset</code> of your particular JRE to find out other supported values.
-un <names>	The named users to dump.
-gn <names>	The named groups to dump
-pn <names>	The named process definitions ("workflows") to dump
-cn <names>	The named content types to dump

Table 3.12. Options of dump

Parameter	Description
<code>-t <paths></code>	The paths of resources to dump.

The standard way to specify a resource to be dumped is its ID. To dump the root folder, for example, you call

```
> cm dump -u admin -p admin 1
```

and get a result like

```
content: coremedia:///cap/content/1
type: Folder (coremedia:///cap/contenttype/Folder_)
path: /
created by: admin (coremedia:///cap/user/0), creation date:
  2004-12-20T07:57:19+00:00
modified by: admin (coremedia:///cap/user/0),
  modification date: 2004-12-20T07:59:48+00:00
is place approved: true, place approved by: -,
  place approval date: 2004-12-20T07:57:19+00:00
is published: true, published by: -, publication date:
  2004-12-20T07:57:19+00:00
to be deleted: false, to be withdrawn: false
is checked out: false, current editor: -
children:
  /Home (coremedia:///cap/content/5)
  /Inbox (coremedia:///cap/content/3)
  /System (coremedia:///cap/content/9)
  /work (coremedia:///cap/content/13)
  /work2 (coremedia:///cap/content/15)
```

The result for a content item looks similar; instead of the children you get detailed information about all versions of the content item. If you don't know the IDs of the resources you want to dump, you can alternatively specify their paths with the `-t` option, for example,

```
> cm dump -u admin -p admin -t /work/article1 /work/image1
```

You can dump users and groups by specifying their names with the `-n` option, for example,

```
> cm dump -u admin -p admin -un publisher -gn system
```

.dumps the user `publisher` and the group `system`.

However, all these examples are only special convenient variants of invoking the `dump` tool. As you can see in the very first line of the result of dumping the root folder, the actual ID of the root folder is not 1, but `coremedia:///cap/content/1`, and this format is the general way to use `dump`:

```
> cm dump -u admin -p admin coremedia:///cap/content/1
```

Each *CoreMedia CMS* object has such an ID and can be dumped. Try the above example for dumping users and groups, and you will get IDs like `coremedia:///cap/user/5` and `coremedia:///cap/group/1`. You will obtain IDs of other objects by the other server tools:

- The tool `cm processdefinitions` gives you IDs of workflow process definitions.
- The tool `cm processes` gives you IDs of running processes.
- [Section “Sessions” \[103\]](#) shows how to get IDs of open sessions.

The dump tool tries some additional heuristics to interpret strings given on the command line that do not match an ID pattern. IDs containing a slash are treated as a content repository path. String may also be interpreted as user names, group names or content type names if possible.

Password Property Encryption

In order to encrypt passwords stored in property files you can use the KeyStore based encryption service. This service uses a pair of public and private keys to encode and decode passwords. The keys are retrieved from a KeyStore located in the file system.

Preliminary Setup

Before you can use the KeyStore based encryption service, you have to create a KeyStore file using the Java `keytool` command. The KeyStore file will contain the private key and the certificate that will be used to encrypt and decrypt the passwords. On the command prompt type:

```
keytool -genkey -alias <KEY_ALIAS> -keyalg RSA -validity 3650
-keystore <KEYSTORE_FILENAME>
```

The tool will ask you for the KeyStore password and the key password.

It will also ask you for your user name, organizational unit, organization, city, state/province, country code. This information (which goes into your self-signed certificate) is not relevant for the correct functioning of your KeyStore. The resulting key/certificate will be valid for 3650 days (about 10 years). It is assumed that this should be enough for your CM installation.

Having the KeyStore created, the KeyStore credentials now have to be stored in a password file, so that the servers and clients can access the KeyStore without prompting for passwords. The password file is in Java properties file format and has to contain the following entries:

```
CM_KEYSTORE_PASSWORD=<KEYSTORE_PASSWORD>
CM_KEY_ALIAS=<KEY_ALIAS>
CM_KEY_PASSWORD=<KEY_PASSWORD>
```

As the password file will contain the clear text passwords for your KeyStore, the file has to be protected from unauthorized access. This could be done for example by setting reasonable access rights for the file, or by putting it on a removable device.

In order to use the KeyStore with the Encryption Service, you have two options:

- By default, the service expects
 - the KeyStore file under the path `${user.home}/.cmservices/.keystore`
 - and the password file under `${user.home}/.cmservices/.keystore.properties`
- If you want store the files under different paths you have to provide the following two system properties:
 - `CM_KEYSTORE_LOCATION`: location of the KeyStore file
 - `CM_KEYSTORE_PASSWORD_FILE_LOCATION`: location of the password file

Password Encryption

For each password you want to encrypt take the following steps:

1. Login as a user who can access the KeyStore and password file. Switch to the installation directory of the command line tools and enter the following command where `<plaintextpassword>` should be replaced with the password you want to encrypt:

```
bin/cm encryptpasswordproperty <plaintextpassword>
```

2. The command output is the encrypted password (which includes the curly brackets!) and some informational text. Use the `-r` option (`bin/cm encryptpasswordproperty -r <plaintextpassword>`) to have the tool just dump out the encrypted password without other information.

The tool will generate a unique value for the same plain text value each time you invoke it.



3. Copy the password (including the curly brackets) into your respective properties file. You can append a comment after the closing curly bracket to add information. For example:

```
sql.store.password={G/7UZ7hPQnGZ/xX4J/7b8FNp/ybEH/JU
Qp5c8NRoDEQS1K5ypbkwotfu6j8U1SHr
QifmKeAQUvou/+ES34/pRHs=} --- generated by User xxx
on 28/03/2013
```

Verify a password

If you want to verify that a given encrypted password actually represents a given plaintext password, use

```
cm encryptpasswordproperty -c <plaintext password> <encrypted entry>
```

where `<plaintext password>` should be replaced with the password and `<encrypted entry>` with the result of the encryption tool. The command will provide you with textual information whether these passwords match. The command returns with return value "0" whether the tokens match or not.

Troubleshooting

View the respective component's log file. If an encrypted password cannot be decoded, you will see an error message in your log file telling you so. Since passwords decryption is verified early on (fail fast), you will find the error messages shortly after the component, service, or server starts.

Events

The `events` utility supports two modes for printing events.

- If the option `-c` is given, the utility prints the events for a single content and exits immediately.
- Otherwise, the utility prints all events generated by the *Content Management Server* or the *Workflow Server* until it is terminated explicitly. At the user's option, a time stamp can be provided for replaying historic events previously to entering the live event stream.

```
usage: cm events -u <user> [other options]
available options:
-c,--content-id <content id>    content id for which the history is
  shown
-d,--domain <domain name>      domain for login (default=<builtin>)
-p,--password <password>      password for login
-t,--timestamp <timestamp>     content timestamp
-u,--user <user name>          user for login (required)
-url <ior url>                  url to connect to
```

Example 3.14. Usage of the events utility

The events utility has the following options:

Table 3.13. Options of the events utility

Parameter	Description
-c	The id of the content whose history is to be printed. Omit this option, if you want to print events for the entire repository. This option cannot be combined with the -t option.
-t	Timestamp, this option gives the date of the last synchronization with events. If you restart the utility, all events which happened after the moment given with the option -t will be reprocessed (that is printed to the console), so -t should be omitted when the events utility is used for the first time. If you start events without the option -t, all events from now on will be printed. A timestamp has the format <seq_no>:<sub_no>:<id_tag> and is printed out before the event. See Javadoc of com.coremedia.cap.content.Timestamp#fromNumbers(int,int,int) for further details. Note that session events don't provide timestamps.

Example 3.15. Output of events

```
Listening from 2147483647:2147483647:0
2005-10-19T11:47:04+02:00: 2224:1:-1057456870: content checked out
on coremedia:///cap/content/1468 by user
coremedia:///cap/user/5 (version coremedia:///cap/version/1468/2)
2005-10-19T11:47:05+02:00: session opened on
coremedia:///cap/session/1268 by coremedia:///cap/user/1 from
armadillo (10.1.4.111) for coremedia:///cap/service/feeder
2005-10-19T11:47:05+02:00: session closed on
coremedia:///cap/session/1268 by coremedia:///cap/user/1 from
armadillo (10.1.4.111) for coremedia:///cap/service/feeder
2005-10-19T11:47:47+02:00: 2225:1:-1057456870: version created on
coremedia:///cap/content/1468 by user
coremedia:///cap/user/5 (version coremedia:///cap/version/1468/2)
2005-10-19T11:47:47+02:00: 2225:2:-1057456870: content checked in
on coremedia:///cap/content/1468 by user
coremedia:///cap/user/5 (version coremedia:///cap/version/1468/2)
```

IOR

With the IOR utility program you can display an IOR in a user readable form. This program determines whether the computer name or an IP is used for communication over CORBA and which CORBA port is used. The CORBA communication settings can be configured (see the [Operations Basics Manual]).

The program is executed with:

```
cm ior <ior1> <ior2> ...
```

Example:

```
> bin/cm ior `wget -qO- http://localhost:44441/coremedia/ior`
...
IOR:000000000000002549444c3a686f782f636f72656d2f636f7262612
f4c6f67696e536572766963653a312e30000000000000001000000000
0000340001000000000016686f78706332342e636f72656d656469612e6
36f6d00042800000010000000003ab08a890005c0a800000000
IOR<IDL:hox/corem/corba/LoginService:1.0><IIOP:1.0:pcname.
coremedia.com:1064>
```

Example 3.16. Execution of CM IOR with IOR of the CM Server

License

The `license` utility prints out the license data of the *CoreMedia CMS* installation for each service.

```
usage: cm license -u <user> [other options] [-v]
available options:
-d,--domain <domain name>    domain for login (default=<builtin>)
-p,--password <password>     password for login
-u,--user <user name>        user for login (required)
-url <ior url>                url to connect to
-v                             verbose
```

Usage of license

The `license` tool has only one additional option:

Parameter	Description
-v	Verbose output, prints out additional information

Table 3.14. Options of license

The standard output of the `license` utility looks like this:

```
license:
licensee: Development-Installation
valid from: 2004-09-30T00:00:00+02:00
valid until: 2005-09-30T00:00:00+02:00
grace until: 2005-12-31T00:00:00+01:00
release: <any>
host: <any>
ip: <any>
workflow customizable: true
service: coremedia:///cap/service/debug, 1/10c, 1/100n,
m<infinity>
service: coremedia:///cap/service/webserver, 0/15c, 0/50n,
m<infinity>
service: coremedia:///cap/service/filesystem, 0/5c, 0/50n,
m<infinity>
service: coremedia:///cap/service/importer, 0/2c, 0/10n,
m<infinity>
service: coremedia:///cap/service/replicator, 0/5c, 0/10n,
m<infinity>
service: coremedia:///cap/service/feeder, 0/2c, 0/10n,
m<infinity>
service: coremedia:///cap/service/publisher, 1/1c, 1/10n,
m<infinity>
service: coremedia:///cap/service/system, 0/5c, 0/25n,
```

```
m<infinity>
service: coremedia:///cap/service/workflow, 1/15c, 1/50n,
m<infinity>
service: coremedia:///cap/service/editor, 1/15c, 1/50n,
m<infinity>
```

The compact notation of the services denotes the used and the available concurrent licenses (such as 1/15c), the used and the available named licenses (such as 1/50n) and the multiplicity (such as m<infinity>). The verbose output of a service additionally shows the users that hold a named license, for example:

```
service: coremedia:///cap/service/editor
concurrent licenses: 15 used: 1
named licenses: 50 used: 1
multiplicity: <infinity>
users:
  rolf (coremedia:///cap/user/102)
```

For details about the license model of the *CoreMedia CMS* see the [Operations Basics Manual].

Processorusage

This utility provides access to data about the usage of processor licenses. For a given time interval, it shows the maximum number of processors used and an explanation of that number in terms of servers and clients that were active at that time. Clients and servers are identified by their IP address. Additionally, it shows the development of the number of processors in use for the given time interval.

Sessions of the *Site Manager* and the *Web Editor* as well as the standard administrative tool are not included. Therefore, only infrastructure processes (servers, web applications, feeders and so on) will appear in the listing. Individual users cannot be identified from the information provided here.

All reported dates are printed in the GMT time zone.

The output of the tool contains an encrypted MD5 hash code. It holds no additional information besides the information that is also presented in clear text.

The tool should normally be run against the *Content Management Server*. It will provide information about all involved machines, even machines of the delivery environment. It is recommended to run the tool regularly in order to detect a shortage of CPU licenses.

```
usage: cm processorusage -u <user name> [options]

available options:

-d,--domain <domain name>  domain for login (default=<builtin>)
-e <days>                  end date for report in days before now
                             (default: 0)
```

Usage of processorusage

```
-i <intervals>          interval count for time series report
                        (default: one interval per day)
-p,--password <password> password for login
-s <days>              start date for report in days before
now                    (default: 365)
-u,--user <user name>  user for login (required)
-url <ior url>          url to connect to
```

Table 3.15. The parameters of `processorusage`

Parameters	Description
-s	The start of the reported interval in days before last midnight. The default is 365 days before the end date.
-e	The end of the reported interval in days before last midnight. Default is last midnight.
-i	The number of time intervals into which the specified interval is split. By default, the interval is split into days.

Example

The call `cm processorusage -u admin` gives you an overview of the last year. To analyze the previous day in more detail, you might want to use `cm processorusage -u admin -s 1 -i 24`.

Publications

The `publications` utility shows all running publications.

Usage of the `publications` utility

```
usage: cm publications -u <user> [other options] [-v]
available options:
-d,--domain <domain name>  domain for login (default=<builtin>)
-p,--password <password>  password for login
-u,--user <user name>      user for login (required)
-url <ior url>              url to connect to
-v                           verbose
```

The `publications` tool has only one additional option:

Table 3.16. Options of the `publications` utility

Parameter	Description
-v	Verbose output, prints out additional information

The verbose output of the `publications` tool looks like this (sample, one publication only):

```
pending publications:
  publication: coremedia:///cap/publication/3
```



```
priority: 60
user: rolf (coremedia:///cap/user/102)
preview: false
size: 4
stage: 4
```

You can also run `publications` without the `-v` option and obtain the details of a publication with the `dump` tool (see [Section “Dump” \[92\]](#)).

Repositorystatistics

You can use the `repositorystatistics` tool to get a simple overview over the content in your *CoreMedia* repository. The tool uses the information in `sql.properties` to connect directly with the database of your *Content Server*. It writes the result to the console. Start the tool with `cm repositorystatistics`. No arguments are required.

You will get information about the following topics:

- Number of users
- Number of groups
- Number of user/group associations
- Number of rules
- Number of folders and children with some statistics
- Number of content items and versions sorted by content type with some statistics

The following example shows the result for a test repository:

```
gathering statistics data
-----
# users: 93
# groups: 19
# user/group associations: 592
# rules: 20

# folders: 19052
# folders in trash: 1256
children: 124084
average children: 7.121442
min children: 1
max children: 1644
< 100: 17348
< 250: 51
< 500: 21
< 1000: 2
< 2500: 2
< 5000: 0
< 10000: 0
>= 10000: 0
```

Example 3.17. Result of repositorystatistics

```
# documents: 131120
# documents in trash: 24831

# doctypes: 31

Book:
documents: 169
versions: 1447
average: 8.56213
min: 1
max: 87
< 10: 117
< 25: 41
< 50: 10
< 100: 1
< 250: 0
< 500: 0
< 1000: 0
>= 1000: 0
```

Rules

The `rules` utility prints out all rules declared for a resource or in the entire repository.

```
usage: cm rules -u <user> [other options]
available options:
-d,--domain <domain name> domain for login (default=<builtin>)
-p,--password <password> password for login
-u,--user <user name> user for login (required)
-url <i>ior url</i> url to connect to
-t, --paths <path> path of contents to dump
-v,--verbose enables verbose output
```

Usage of the rules utility

`Rules` has one additional option.

Parameter	Description
<code>-t <paths></code>	Enter the paths of resources for which you want to dump the attached rules.

Table 3.17. Options of the rules utility

The output of the `rules` utility looks like this (sample, one rule only):

```
content: / (coremedia:///cap/content/1)
group: editor (coremedia:///cap/group/2)
type: coremedia:///cap/contenttype/Document_
rights: RMD
```

For details about the meaning of rules see [Section 3.16.2, “User Rights Management” \[163\]](#).

Sessions

The `sessions` utility shows all open sessions.

```
usage: cm sessions -u <user> [other options] [-v]
available options:
-d,--domain <domain name> domain for login (default= <builtin>)
-p,--password <password> password for login
-u,--user <user name> user for login (required)
-url <ior url> url to connect to
-v verbose
```

Usage of the sessions utility

The `sessions` tool has only one additional option:

Parameter	Description
-v	Verbose output, prints out additional information

Table 3.18. Options of the session utility

The verbose output of the `sessions` utility looks like this (sample, one session only):

```
session: coremedia:///cap/session/11
privileged: false
user: rolf (coremedia:///cap/user/9)
client type: GUI Editor
host: nightshade (0.0.0.1)
started at: 2004-12-21T11:10:37+01:00
service: coremedia:///cap/service/editor
```

Systeminfo

The `systeminfo` tool prints out information about the *CoreMedia CMS* installation itself. This includes especially the version which you will need whenever you request support.

```
usage: cm systeminfo -u <user> [other options]
available options:
-d,--domain <domain name> domain for login (default=<builtin>)
-p,--password <password> password for login
-u,--user <user name> user for login (required)
-url <ior url> url to connect to
-v,--verbose enables verbose output
```

Usage of systeminfo

That is, `systeminfo` has none but the standard options.

The output of `systeminfo` covers the client environment in which `systeminfo` itself is running, the content server and (if available) the workflow server. It contains information about the *CoreMedia CMS* versions, the JVM versions, the operating systems, the occupied ports and some configuration details.

Validate Multi-Site

The tool `validate-multisite` helps you to manage multiple localized variants of contents in *CoreMedia Digital Experience Platform 8* as described in chapter "CoreMedia Digital Experience Platform 8 Website and Content Structure/Localized Content Management" of the [CoreMedia Digital Experience Platform 8 Developer Manual]. It validates your multi-site configuration as well as your content repository.

It is strongly recommended that you run this tool when introducing or migrating the multi-site feature to ensure that your system is valid. You can also use this tool to regularly check your content for possible mistakes during translation processes, checking global conditions that are not addressed by the validators that are available to your editors in *CoreMedia Studio*.

```
cm validate-multisite [ {-b folder} ...] [-f file] [-fow] [other...]
```

Example 3.18. Usage of validate-multisite

Table 3.19. Parameters of validate-multisite

Parameter	Description
{-b --below } <i>folder</i>	Base folder for validating content. May be used multiple times to validate multiple folders. By default, all sites as located by their site indicator are validated. If you include folder outside of any site, the analysis results will not be meaningful, because most conditions cannot be checked in such a case. As long as a site indicator is invalid, it cannot define a site, so that analyzing content in the site will not be possible.
{-f --file } <i>file</i>	Output file for writing the detected issues in tab-separated value format.
-fow --fail-on-warning	Fail with a non-zero exit code when a warning is generated.

Severities

The multi-site issues come with different severities. It is recommended to fix all issues for best results when using multi-site.

- INFO** denotes issues that oppose best practices
- WARN** denotes recoverable issues which violate multi-site requirements and possibly cause unexpected results
- ERROR** denotes unrecoverable issues which prohibit further analysis as well as using multi-site features will not work or produce corrupt results

File Output

The file output is meant to support automatic actions like for example adjusting wrong or missing property values. It contains relevant error parameters in tab separated columns, so that it becomes possible to identify the reported issues.

The first two columns are fixed: Issue Severity and Issue Error Code. The third is most of the time the id of the analyzed content. Additional parameters are mentioned in [Table 3.20, “Issues of validate-multisite” \[105\]](#).

Issues

Table 3.20. Issues of validate-multisite

MS-VALIDATION-0000 - Internal Error	
Severity	ERROR
Description	An exception occurred during validation. Please analyze the exception stack trace for further information.
File Output	3. Exception Message
MS-VALIDATION-1000 - Missing Sites Service	
Severity	ERROR
Description	SitesService is not defined. None of the multi-site features will work.
File Output	<i>no additional columns</i>
Option(s)	Validate your application contexts to contain a bean of type SitesService. Typically, by adding a dependency to <code>com.coremedia.cms:cap-multisite</code> and importing <code>multisite-services.xml</code> . <pre><import resource="classpath:/com/coremedia/cap/multisite/multisite-services.xml"/></pre>
MS-VALIDATION-1001 - Missing Site Model	
Severity	ERROR
Description	SiteModel is not available in SitesService. None of the multi-site features will work.
File Output	<i>no additional columns</i>
Option(s)	Validate your application contexts to contain a bean of type SitesService with reference to a SiteModel. Typically, by adding a dependency to <code>com.coremedia.cms:cap-multisite</code> , importing <code>multisite-services.xml</code> and adding your configured SiteModel bean.

	<pre data-bbox="275 224 887 354"><import resource="/classpath:/com/coremedia/cap/multisite/multisite-services.xml"/> <customize:replace id="addYourSiteModel" bean="sitesService" property="siteModel" custom-ref="yourSiteModel"/></pre> <p data-bbox="275 378 887 451">It is recommended to create a bean of the type <code>com.coremedia.cap.multisite.DefaultSiteModel</code> or of a custom subtype.</p>
MS-VALIDATION-2000 - Missing Required Property Value	
Severity	WARN, ERROR
Description	Denoted property's value is empty or unset but is either required (error) or strongly recommended (warn).
File Output	3. name of property
Option(s)	Set the required/recommended property value.
MS-VALIDATION-3000 - Missing Site Indicator Content Type	
Severity	ERROR
Description	Content type defined in property <code>siteIndicatorDocumentType</code> of site model does not exist.
File Output	3. referenced non-existing content type
Option(s)	Either update site model configuration or extend your content type model by the given type.
MS-VALIDATION-3001 - Missing Property Descriptor	
Severity	ERROR
Description	Property referenced from site model does not exist.
File Output	3. content type which does not serve defined property 4. name of missing property 5. site model property referencing missing property
Option(s)	Either update site model configuration or extend your content type model by the required property.
MS-VALIDATION-3002 - Invalid Property Descriptor Type	
Severity	ERROR
Description	Property referenced from site model exists but is of invalid type. For example the denoted <code>master</code> property is not of type link.

File Output	<ol style="list-style-type: none"> 3. content type which contains property 4. name of property 5. site model property referencing property 6. expected property type 7. actual property type
Option(s)	Either update site model configuration or change your content type model to match the required type.
MS-VALIDATION-3003 - Master Link not Weak	
Severity	WARN
Description	The master property is not marked as weak. If you do not fix this issue, you always have to publish the master content before its derived contents.
File Output	<ol style="list-style-type: none"> 3. affected content type 4. affected property
Option(s)	Adjust your content type model setting master property to weak. <pre><LinkListProperty Name="master" Max="1" LinkType="CMLocalized" extensions:weakLink="true"/></pre>
MS-VALIDATION-3004 - Missing Localizable Content Type Property	
Severity	INFO
Description	A content type appears to be localizable but misses a recommended property.
File Output	<ol style="list-style-type: none"> 3. content type 4. property
Option(s)	Adjust your content type model to match the required criteria.
MS-VALIDATION-3005 - Invalid Localizable Content Type Property	
Severity	WARN
Description	A content type appears to be localizable but one of its properties is of invalid type.
File Output	<ol style="list-style-type: none"> 3. content type 4. property 5. expected type 6. actual type

Option(s)	Adjust your content type model to match the required criteria.
MS-VALIDATION-4000 - Invalid Property Value	
Severity	WARN
Description	Property value is invalid.
File Output	<ul style="list-style-type: none"> 3. site indicator's content id 4. name of property 5. actual value 6. suggested value for fix
Option(s)	Fix the property value.
MS-VALIDATION-4001 - Ambiguous Property Value	
Severity	INFO
Description	Property value is ambiguous and might lead to unexpected behavior.
File Output	<ul style="list-style-type: none"> 3. site indicator's content id 4. name of property
Option(s)	Fix the property value.
MS-VALIDATION-4002 - Invalid Path	
Severity	WARN
Description	A site indicator is located at a path that does not match the configured site root folder pattern.
File Output	<ul style="list-style-type: none"> 3. content id 4. expected path; might be some root folder rather than explicit path
Option(s)	Either fix the content location or adjust the site model to match the location of the content.
MS-VALIDATION-4003 - Invalid Path	
Severity	ERROR
Description	A site indicator is located a path that does not match the currently configured site root folder pattern and site indicator depth.
File Output	<ul style="list-style-type: none"> 3. content id 4. configured depth 5. actual depth 6. folder content should be located below at configured depth

Option(s)	Either fix the content location or adjust the site model to match the location of the content.
MS-VALIDATION-4004 - No Site Indicators	
Severity	WARN
Description	No site indicators found. Either you need to adjust your site model or your content needs to have site indicators. Not detected when folder restriction is active.
File Output	<i>no additional columns</i>
Option(s)	Either fix the content location or adjust the site model to match the location of the content.
MS-VALIDATION-4005 - Multiple Site Indicators at Site Root	
Severity	ERROR
Description	Multiple site indicators exist in the same site at the same depth as denoted by site indicator depth in site model. Selecting and deriving sites might cause problems. Possibly not detected when folder restriction is active.
File Output	3. first site indicator's id 4. second site indicator's id 5. folder which both of them identify as site folder
Option(s)	Remove unnecessary site indicators.
MS-VALIDATION-4006 - Non-Unique Site-ID	
Severity	ERROR
Description	Multiple site indicators share the same site ID. Possibly not detected when folder restriction is active.
File Output	3. first site indicator's id 4. second site indicator's id 5. id which both of them share
Option(s)	Create unique site IDs as administrative user.
MS-VALIDATION-4007 - Missing Site Indicator Property Value	
Severity	ERROR
Description	The denoted property value of a site indicator is empty or unset but is required.
File Output	3. affected site indicator's id

	4. name of property
Option(s)	Set the required property value.
MS-VALIDATION-4008 - Root not in Site	
Severity	WARN
Description	The root document referenced by site indicator is outside the site folder. This will cause problems when deriving a new site.
File Output	3. referencing site indicator's id 4. site's id the site indicator belongs to 5. root document's id not in site
Option(s)	Move root document somewhere into the site folder.
MS-VALIDATION-5000 - Type not Localizable	
Severity	WARN
Description	A content inside a site is not of a localizable content type. This might cause issues in translation process.
File Output	3. affected content id 4. actual content type
Option(s)	Most likely adjust your content type model so that the content's type is localizable.
MS-VALIDATION-5001 - Not Site Locale	
Severity	WARN
Description	A content has locale that differs from the site locale, but is not marked as derived from another content in the site.
File Output	3. affected content id 4. locale of content 5. expected site locale
Option(s)	Most likely adjust locale or set a master link.
MS-VALIDATION-5002 - Locale not Set	
Severity	INFO
Description	A content is localizable, but the locale property is not set. In this case, the locale defaults to the site locale, but it is preferred to set the locale explicitly.
File Output	3. affected content id

	4. suggested locale from site
Option(s)	Set the locale of the given content to the site locale.
MS-VALIDATION-5003 - Same Locale	
Severity	WARN
Description	A content is marked as derived from another content in the site, but still uses the site locale.
File Output	3. affected content id 4. actual locale
Option(s)	Most likely adjust the locale of the derived content.
MS-VALIDATION-6000 - Content not Used in Derived Site	
Severity	INFO
Description	A content in the master site has no translated counterpart in the derived site. This might be a result of an unfinished translation process as well as intended as content is not required for the derived site.
File Output	3. affected content id 4. content's site id 5. locale of missing translation
Option(s)	Possibly nothing to do. Otherwise, start a translation process to get the content from master into derived site.
See Also	<ul style="list-style-type: none"> ➔ MS-VALIDATION-6001 - Content not Used in Derived Site but some Content Exists [111] ➔ MS-VALIDATION-6002 - Content without Master [112]
MS-VALIDATION-6001 - Content not Used in Derived Site but some Content Exists	
Severity	WARN
Description	A content in master site has no translated counterpart in the derived site regarding the value of master link. Nevertheless, a content with the very same name exists in derived site.
File Output	3. affected content id 4. content's site id 5. locale of missing translation 6. content id of suggested counterpart to set current content as master

Option(s)	Most likely just set the master link or rename preferably the derived document to prevent future collisions in translation process.
See Also	<ul style="list-style-type: none"> → MS-VALIDATION-6000 - Content not Used in Derived Site [111] → MS-VALIDATION-6003 - Content without Master but some Content Exists [112]
MS-VALIDATION-6002 - Content without Master	
Severity	INFO
Description	Counterpart to MS-VALIDATION-6000 [111] : In this case a content in the derived site exists which has no master. This might be intended if you have a content which is just relevant for your derived site.
File Output	<ul style="list-style-type: none"> 3. affected content id 4. content's site id
Option(s)	Possibly nothing to do. Otherwise, copy (and translate) content to master site and add a corresponding master link to the derived content.
See Also	<ul style="list-style-type: none"> → MS-VALIDATION-6000 - Content not Used in Derived Site [111] → MS-VALIDATION-6003 - Content without Master but some Content Exists [112]
MS-VALIDATION-6003 - Content without Master but some Content Exists	
Severity	INFO
Description	Counterpart to MS-VALIDATION-6001 [111] : In this case a content in the derived site exists which has no master link set but a content with the same name exists in master site.
File Output	<ul style="list-style-type: none"> 3. affected content id 4. content's site id 5. content id of suggested counterpart to as master
Option(s)	Most likely just set the master link or rename preferably the derived document to prevent future collisions in translation process.
See Also	<ul style="list-style-type: none"> → MS-VALIDATION-6001 - Content not Used in Derived Site but some Content Exists [111] → MS-VALIDATION-6002 - Content without Master [112]
MS-VALIDATION-6004 - Root Content with Master	
Severity	INFO

Description	A content in a root site defines a master.
File Output	3. affected content id 4. current site id 5. linked master content id
Option(s)	Possibly delete the master link or adjust site's master link.
MS-VALIDATION-6005 - Master not in Site	
Severity	WARN
Description	A content in a derived site is derived from a content that belongs neither to the master site nor the content's site.
File Output	3. affected content id 4. current site id 5. linked master content id 6. master site id
Option(s)	Adjust the master link to point to either to a content inside the site or in the master site.
MS-VALIDATION-6006 - Master has more than one derived variant in one site	
Severity	WARN
Description	A master content has more than one directly derived variant in one single site. This harms the translation process and leads to inconsistent translation results.
File Output	3. affected content id 4. current site id 5. locale of the affected site with the two or more derived variants
Option(s)	Check if every derived variant has the correct master set. Reduce to only one content, which refers to the master.
MS-VALIDATION-7000 - Master Version Destroyed	
Severity	WARN
Description	A master version referenced by derived content does not exist any longer.
File Output	3. affected content id 4. master content id 5. denoted destroyed master version 6. the latest master version

Option(s)	Either adjust master version or start a translation process to update master version.
See Also	<ul style="list-style-type: none"> ➔ MS-VALIDATION-7001 - Master Version Invalid [114] ➔ MS-VALIDATION-7002 - Master Version not Localized yet [114] ➔ MS-VALIDATION-7003 - Master Version not up to Date [115]
MS-VALIDATION-7001 - Master Version Invalid	
Severity	WARN
Description	A master version value is invalid regarding the available versions of the master. Number might be negative or greater than current version number.
File Output	<ul style="list-style-type: none"> 3. affected content id 4. master content id 5. denoted invalid master version 6. the latest master version
Option(s)	Either adjust master version or start a translation process to update master version.
See Also	<ul style="list-style-type: none"> ➔ MS-VALIDATION-7000 - Master Version Destroyed [113] ➔ MS-VALIDATION-7002 - Master Version not Localized yet [114] ➔ MS-VALIDATION-7003 - Master Version not up to Date [115]
MS-VALIDATION-7002 - Master Version not Localized yet	
Severity	INFO
Description	Derived content has not been localized yet. Because this is a completely valid state, it is reported with severity INFO.
File Output	<ul style="list-style-type: none"> 3. affected content id 4. master content id 5. the latest master version
Option(s)	Start a translation process to update master version.
See Also	<ul style="list-style-type: none"> ➔ MS-VALIDATION-7000 - Master Version Destroyed [113] ➔ MS-VALIDATION-7001 - Master Version Invalid [114] ➔ MS-VALIDATION-7003 - Master Version not up to Date [115]

MS-VALIDATION-7003 - Master Version not up to Date	
Severity	INFO
Description	Derived content has not been updated to most recent master version yet. Because this is a completely valid state, it is reported with severity INFO.
File Output	3. affected content id 4. master content id 5. denoted outdated master version 6. the latest master version
Option(s)	Start a translation process to update master version.
See Also	→ MS-VALIDATION-7000 - Master Version Destroyed [113] → MS-VALIDATION-7001 - Master Version Invalid [114] → MS-VALIDATION-7002 - Master Version not Localized yet [114]
MS-VALIDATION-8000 - Link Spanning Sites	
Severity	WARN
Description	A content in a site links to a content in another site with a property other than the master property.
File Output	3. affected content id 4. link property name containing spanning links
Option(s)	Possibly adjust the links.
MS-VALIDATION-8001 - Link to Destroyed Content	
Severity	WARN
Description	A content links to a non-existing content.
File Output	3. affected content id 4. link property containing link to destroyed content
Option(s)	Adjust the link.
MS-VALIDATION-8002 - Link to Deleted Content	
Severity	WARN
Description	A content links to a deleted content.
File Output	3. affected content id 4. link property containing link to deleted content

Option(s)	Adjust the link.
-----------	------------------

Exit Codes

The tool exits with the following codes:

- Code 0** No issues other than severity INFO (or WARN if fail on warning is turned off) found.
- Code 31** Errors found.
- Code 32** Warnings found (if fail on warning is turned on).
- other** Other exit code are standard to UAPI Client and denote exceptions, usage failure, etc.

3.14.2 Operation

This section describes the operational server utilities. Before you apply these tools, you should read the respective documentation thoroughly. The operational tools perform highly privileged tasks some of which are not subject to any consistency checks. An improper usage of the tools may severely affect the performance of the CoreMedia CMS or even damage the repository.

Operational tools

Repository tools

- `cleanrecyclebin/cleanversions`: Delete obsolete content items and versions from the repository
- `serverexport`: Stores the content of the repository as XML files
- `serverimport`: Upload content stored by `serverexport`
- `usedlicenses`: Withdraws a license from a user
- `groovysh`: The *Groovy Shell* lets you interactively access the *Content Server* using the *Unified API*.

Database tools

- `sql`: Provides direct access to the database
- `dbindex`: Creates a database index over the folder structure

Server tools

- `cancelpublication`: Cancel a running publication

- `killsession`: Kills a session
- `tracesession`: Causes a session to be traced
- `runlevel`: Changes the runlevel of the server
- `unlockcontentserver`: Remove a lock that stops a *Content Server* from starting up

User tools

- `changepassword`: Changes the password.
- `dumpusers`: Writes an XML file containing a representation of the users and groups defined in the internal user administration, and of all rights rules.
- `restoreusers`: Updates the internal user administration and the rights rules from an XML file containing a representation of users, groups and rules.

Note that some of the operational tools do not access the *CoreMedia CMS* server but work on database level. Those tools don't support the standard options because they don't need a user to open a session.

Cancelpublication

The `cancelpublication` tool can be used to cancel long-running publications that prevent users from publishing other content. A publication that should be canceled has to be in on of the stages:

- `queued`
- `checking preconditions`
- `scheduling`
- `checking dependencies`

When the publication has reached the stage started, that means it is running, it is not possible to cancel it anymore.

Long-running publications can occur when you have a complex content model structure with many links. You publish a deleted content item that is linked with many other content items, for example. This would lead to a long running dependencies checking stage which you can cancel with the tool.

So this is an emergency tool which you will need in exceptional cases only.

You can obtain the IDs of the running publications with the `publications` utility (see [Section "Publications" \[100\]](#)).

Usage of cancelpublication

```
usage: cm cancelpublication -u <user> [other options]
      <publicationId> ...
available options:
-d,--domain <domain name>    domain for login (default=<builtin>)
-p,--password <password>    password for login
-u,--user <user name>       user for login (required)
-url <ior url>              url to connect to
-v,--verbose                 enables verbose output
```

Changepassword

The `changepassword` tool changes the password of a user. An administrator can change the password of every other user, all other users may only change their own passwords.

Example 3.19.

```
usage: cm changepassword -u <user> [other options]
      [-n <user name>] -w <new password>
available options:
-w,--new-password <new password>    new password for the given
                                      user or the login user, if no
                                      user was given
-d,--domain <domain name>           domain for login
                                      (default=<builtin>)
-n,--name <user name>              name of the user to change
                                      the password for
                                      (administrators only)
-p,--password <password>           password for login
-u,--user <user name>              user for login (required)
-url <ior url>                      url to connect to
-v,--verbose                        enables verbose output
```

The options have the following meaning:

Table 3.21. Options of the changepassword tool

Parameters	Description
-n	Name of the user for whom the password should be changed. This option is only available for administrators. For other users the user name will be taken from the -u option.
-w	New password for the user.

Cleaners

The *CoreMedia Cleaners* are tools for freeing up memory on the server and the underlying database. They are command line tools with no graphical user interface and are intended for use by system administrators only. Administrators are advised to run these collectors regularly (by cron jobs, for example) in order to maximize efficiency and speed of the repository. They come in two versions:

- **Clean Recycle Bin:** A deleted resource is moved into the *Recycle Bin*. To remove the resource from the *Recycle Bin* use the *Clean Recycle Bin* utility. It will remove a resource A permanently from the *Recycle Bin* if and only if there is no resource B outside the *Recycle Bin* that is linked to A. This ensures that no dead links exist in the system (or on a website that is based on the content). On high-performance systems this collector should be run once a week.
- **Clean Versions:** The *Clean Versions* destroys unnecessary intermediate versions of content items. Flexible parametrization allows for arbitrary version selection. The cleaner will not alter the servers runlevel. By providing some folders which are then processed recursively it is possible to divide the cleaning workload across multiple collection jobs.

Both types of cleaners are closely tied - although the actual processing is quite different. Imagine two types of content items: `Navigation` and `Article`. `Navigation` content items point to `Articles` with some `LinkList` Property. `Articles` change daily and are linked to a single navigation content item. As long as some version of the navigation content item still points to an article, this article cannot be collected out of the *Recycle Bin*. The solution is to run the *Clean Versions* in order to delete the intermediate versions of the navigation content item. This will also remove the old links to articles in the *Recycle Bin* - and thus allow the content item collector to finally remove these content items from the *Recycle Bin*.

This is the reason why *Clean Recycle Bin* and *Clean Versions* should be run regularly. Both types of cleaners run in a distributed system and are processing and IO intensive. Actual running time depends heavily on network latencies, database performance and number of resources processed.

Clean Recycle Bin

Before any content item is permanently removed from the recycle bin, the XML representation is written ("exported") to local filesystem. You may use a customized version of the *XML Importer* to import these content items later. If you are sure, that you don't need these files anymore, you can use the option `--noexport` to suppress the export.

When you start **cleanrecyclebin** without other options than the connection parameters, only content will be removed that was last modified 30 or more days ago (see the `--before` parameter).



Usage

Starts `cleanrecyclebin` in standard mode.

```
bin/cm cleanrecyclebin <connection parameters>
  [--directory <a>] [--export-delay <b>]
```

```
[--export-threads <c>] [--export-to-subdirs <d>]
[--noexport] [--after <e>] [--before <f>]
```

Starts `cleanrecyclebin` in simulation mode. This means that no documents will be destroyed nor anything will be exported.

```
bin/cm cleanrecyclebin <connection parameters>
--simulate [--after <a>] [--before <b>]
```

Cancels a currently running `cleanrecyclebin` process.

```
bin/cm cleanrecyclebin <connection parameters> --cancel
```

Table 3.22. Options of the clean recycle bin

Parameter	Description
<code>--simulate</code>	Enables simulation mode
<code>--after</code>	Only remove content items with modification date after or equal to the given date. Format: <code>yyyyMMddHHmmss</code> (such as, 20050102140000). Default is "start of time" in the Unix world.
<code>--before</code>	Only remove content items with modification date before or equal to the given date. Format: <code>yyyyMMddHHmmss</code> . Default is 30 days from the actual date.
<code>--directory</code>	This options specifies the directory where the content items will be exported before destruction. Default is <code>./destroy</code> .
<code>--export-delay</code>	Delay in milliseconds between the export of each content item. Default is "0".
<code>--export-threads</code>	Concurrent threads to use for collecting and exporting. Default is "1".
<code>--noexport</code>	No XML files will be generated if this option is set. By default, XML files will be generated.
<code>--export-to-subdirs [creationDate modificationDate exportDate]</code>	Defines which date should be used to create the directory hierarchy. The hierarchy is build in the way Year<Month<Day<Hour. That is, you have four directories below the main export directory. "Hour" directories are only created, if the "exportDate" is used. Default is a flat directory structure.
<code>--cancel</code>	This will cancel the execution of the document collector. This can cause dead links! Use with care.

Example

In order to remove all content items from the recycle bin that are not referenced by any other content item outside the recycle bin and have a modification date that is less than or equal to 2001-12-31 call:

```
cm cleanrecyclebin --user admin --password admin --simulate --before 20011231000000
```

Performance

On systems with many content items in the recycle bin the collection process might take some time to run its mark and sweep phase. If the process seems to do nothing (it generates no log output), don't kill it. This phase might take an hour or more.

Clean Versions

Run Clean Versions using the following command line:

```
usage: cm cleanversions -u <user> <other options> [--simulate]
      [--export]
      [--directory <directory>]
      [--export-to-subdirs <mode>]
      [--keep-number <n>]
      [--keep-days <d>]
      [--published-only | --unpublished-only]
      [--approved-only | --unapproved-only]
      [-R ] [-l <limit> ] [ <idl>... | -I <idl>... |
      -t <path1>... | -cq <query1>... ]

- OR -

cm cleanversions -u <user> <other options> [--simulate]
      [--export]
      [--directory <directory>]
      [--export-to-subdirs <mode>]
      --selector-class <classname>
      [--selector-param <name:value1>]
      [-R ] [-l <limit> ] [ <idl>... | -I <idl>... |
      -t <path1>... | -cq <query1>... ]
```

available options:

-ao,--approved-only	clean approved versions only
-c,--cache <arg>	size of cache (in bytes) (default=64MB)
-cq,--contentquery <query>	query/queries to select contents
-d,--domain <domain name>	domain for login (default=<builtin>)
-dir,--directory <arg>	directory where document shall be exported <default=destroy>
-es,--export-to-subdirs <arg>	structure of export subdirectories (default=creationDate). One of [base creationDate modificationDate exportDate]
-I,--id <id>	id(s) to select content objects
-kd,--keep-days <arg>	minimum number of days after last modification <default=7>

<code>-kn,--keep-number <arg></code>	minimum Number of versions to keep (default=1)
<code>-l,--limit <limit></code>	limits the number of content objects
<code>-lc,--selector-class <arg></code>	to select, negative for unlimited which is also the default custom VersionSelector implementation class
<code>-lp,--selector-param <arg></code>	custom VersionSelector parameter
<code>-p,--password <password></code>	password for login
<code>-po,--published-only</code>	clean published versions only
<code>-R,--recyclebin</code>	include contents from recycle bin
<code>-s,--simulate</code>	simulation mode where no versions will be destroyed
<code>-t,--path <path></code>	path(s) to select content objects
<code>-u,--user <user name></code>	user for login (required)
<code>-uao,--unapproved-only</code>	clean unapproved versions only
<code>-upo,--unpublished-only</code>	clean unpublished versions only
<code>-url <ior url></code>	url to connect to
<code>-v,--verbose</code>	enables verbose output
<code>-x,--export</code>	export destroyed versions as xml

Clean Versions is applied to every content selected via the content selection parameters. If folders are selected the clean operation is performed recursively on every content item inside.

No matter which versions you select by command line arguments or by custom VersionSelector (see below), some versions will never be cleaned up, which are:

- ➔ the working version,
- ➔ the latest published version,
- ➔ the latest checked in version and
- ➔ for multi-site set ups: the versions which are referred to as master from the latest versions of the referrers.
- ➔ the latest version sent into a translation workflow

Parameter	Description
<code>--keep-number</code>	Minimum number of versions to keep (default = 1). This option precedes all other options. Will be ignored for custom version selectors.
<code>--keep-days</code>	Do only destroy versions that are at least the given number of days older than the last modification date of the content (default = 7 days). Will be ignored for custom version selectors.

Table 3.23. Switches of the version collector

Parameter	Description
<code>--published-only</code> <code>--unpublished-only</code>	Clean published or unpublished versions only. These options can not be used together. If none of them is used, both types will be deleted. Keep in mind, that the latest published version will never be deleted by this tool. Will be ignored for custom version selectors.
<code>--approved-only</code> <code>--unapproved-only</code>	Clean approved or unapproved versions only. These options can not be used together. If none of them is used, both types will be deleted. Will be ignored for custom version selectors.
<code>--simulate</code>	Do not destroy any version but perform a simulation.
<code>--cache</code>	Capacity of cache in Bytes. Default is 64MB.
<code>--selector-class</code>	The class name of a custom <code>VersionSelector</code> (implementing <code>com.coremedia.cotopaxi.util.VersionSelector</code>). Mind that using a custom <code>VersionSelector</code> will ignore some of the parameters stated above, because the custom selector replaces the default selection criterias. Use <code>--selector-param</code> instead to feed parameters to your custom selector.
<code>--selector-param</code>	An optional parameter that will be passed to the custom <code>VersionSelector</code> .
<code>--export</code>	Export versions to file system as XML representation before destroying them.
<code>--directory</code>	The directory where the content items will be exported before destruction. Default is <code>./destroy</code> .
<code>--export-to-subdirs [creationDate modificationDate exportDate]</code>	Defines which date should be used to create the directory hierarchy. The hierarchy is build in the way Year/Month/Day/Hour. That is, you get up to four directories below the main export directory. "Hour" directories are only created if the option "exportDate" is used. Default is a flat directory structure.
<code>-cq <query></code>	Content Selection Content Query to locate contents. Parameter can be used multiple times. Results will be ORED. Clean Versions will traverse folders recursively.

Parameter	Description
<code>-R</code>	Content Selection Add contents from recyclebin. Could also be done via <code>-cq</code> .
<code>-I <id></code>	Content Selection Select contents via explicit id. This is an alternative if you want to mix explicit ids with for example content queries. Parameter can be used multiple times. Results will be ORed. <code>Clean Versions</code> will traverse folders recursively.
<code>-t <path></code>	Content Selection The path of a content to be cleaned. Parameter can be used multiple times. Results will be ORed. <code>Clean Versions</code> will traverse folders recursively.
<code><ids></code>	Content Selection Content IDs to be cleaned. If mixing with other content selection parameters it is recommended to explicitly use the parameter <code>-I</code> Parameter can be used multiple times. Results will be ORed. <code>Clean Versions</code> will traverse folders recursively.
<code>-l <limit></code>	Content Selection Limit the number of selected contents. Default is unlimited. Limit does not apply to recursively located contents.

For information about the UAPI query syntax see Section 5.5, “Query Service” in *CoreMedia Unified API Developer Manual*.



Example for calling *Clean Version* to simulate the cleaning of all content items beneath the root folder but the last two versions:

```
cm cleanversions -u admin -p admin --simulate
  --keep-number 2 --keep-days 0 --path /
```

Example for calling *Clean Version* on all documents despite the `/Home` folder:

```
cm cleanversions -u admin -p admin
  -cq "TYPE Document_ : NOT BELOW PATH '/Home'"
```

Example for calling *Clean Version* to delete all versions below the folders `/Pictures`, `/Articles` and in the recycle bin that are older than 10 days compared to the last modification date but keep at least 3 versions:

```
cm cleanversions -u admin -p admin --keep-number 3
  --keep-days 10 --recyclebin --path /Pictures /Articles
```


Example for calling *Clean Version* to delete all versions below the root folder that are older than 7 days compared to the last modification date (default value) and that are published or approved:

```
cm cleanversions -u admin -p admin --published-only --approved-only
--path /
```

Example for calling *Clean Version* using a custom VersionSelector on resource with id "135"

```
cm cleanversions -u admin -p admin
--selector-class my.package.TestVersionSelector
--selector-param "some parameter" 135
```

Performance and Workload Sharing

This tool is implemented as a remote client, and needs to load each visited content item from the content server. This causes network traffic, and puts considerable load on the server, which means that cleaning will not be fast. To avoid obstruction of the regular editorial work, the tool should be run during off-hours (at night, for instance). The workload should be partitioned so that each cleaning run finishes after a sensible time. Try running the `clean versions` every night and always give it a different folder to work on.

Custom VersionSelectors for cleaning

Using the *Unified API* it is possible to write your own selector mechanism for selecting content item versions to be cleaned. Given a list of versions of a single content item, the selector returns all versions to be destroyed. The behavior of your class may be adjusted using command line arguments, which are passed to its constructor.

Keep in mind, that a custom version selector replaces the default selection, it does not extend it.



The named class must be a public class implementing `com.coremedia.coto.paxi.util.VersionSelector`. It must either define a public no-args constructor, or a public constructor with only one argument of type `java.lang.String`. In the latter case, a `String` containing an additional parameter will be passed to the constructor. The simplest example of a customized predicate:

```
package my.package;

import com.coremedia.cotopaxi.util.VersionSelector;
import com.coremedia.cap.common.CapException;
import com.coremedia.cap.content.Version;
import java.util.ArrayList;
import java.util.List;
```

Example 3.20. Example of a customized predicate

```
public class TestVersionSelector implements VersionSelector {
    public TestVersionSelector(String parameter) {
        System.out.println("Parameter: "+parameter);
    }

    public List selectVersions(List versions) throws CapException {
        List result = new ArrayList();
        for( int x=0; x<versions.size(); x++ ) {
            Version version = (Version) versions.get(x);
            result.add(version);
        }
        return result;
    }
}
```

DBIndex

The *dbindex* utility is intended for creating an index over the folder structure of your *CoreMedia CMS* repository. This index will enhance the speed of a query using the `descendantOf` criterion. However, maintaining this index may affect the performance of the server, especially if you restructure large folders frequently. There is no need to create the index if `descendantOf` queries are used rarely or the folder tree is small.

You cannot access the *CoreMedia CMS* repository for some minutes after start of the utility.

A log message like this "Warning: cap.server.store: SQL Query: finding candidate folders for query takes a long time (5008ms for 348 folders so far); consider activating the folder index" is a good pointer, that you should use the *dbindex* tool.



Usage of dbindex

```
usage: cm dbindex -u <user> [other options] [ --create | --drop |
      --rebuild | --enable | --disable ]
available options:
-b,--rebuild          rebuild index
-c,--create           create index
-r,--drop            drop index
-e,--enable          enable index
-i,--disable         disable index
-p,--password <password> password for login
-d,--domain <domain name> domain for login (default=<builtin>)
-u,--user <user name> user for login (required)
-url <i>or url</i>    url to connect to
-v,--verbose         enables verbose output
```

The options have the following meaning:

Table 3.24. Options of dbindex

Parameter	Description
-b	Rebuild the existing index. You need this only in exceptional cases. Normally you will create or enable the index.
-c	Create the index for the first time or after you dropped it.
-r	Drop the index.
-e	Enable the index after you temporarily disabled it. After major changes in the folder structure it might be faster to drop and create the index.
-i	Temporarily disable the index. You might want to do so for some fast changes in the folder structure.

After changes in the folder structure have been made, the index is automatically updated while the content server is online. If a large portion of the folder structure changes, that is several moves of large sub trees have been made, this may take a moment. So, there can be a short delay in `descendantOf` queries afterwards.

To avoid unnecessary updates of the database, the index on a Live System (*Master Live Server* or *Replication Live Server*) should only be enabled if there are `descendantOf` queries from inside the CAE JSP templates.

Dumpusers

The `dumpusers` tool writes all groups, users and rules currently managed in a *Content Server* into an XML file. You can use this file later on with the `restoreusers` tool to restore the user settings. The structure of the XML file is defined by the `lib/xml/coremedia-userrepository.xsd` schema. It contains a nested structure of group elements which contain rules defined for this group and elements representing the members of this group.

Groups managed in an external user repository only appear in the output file if there are rights rules defined for those groups.

Example 3.21.

```
usage: cm dumpusers -u <user> [other options] [[-e <encoding>]
-f <dump-file-name>]

available options:

-d,--domain <domain name>    domain for login (default=<builtin>)
-e,--encoding <encoding>    encoding for the output
-f,--file <file>            file name of the user repository dump
-p,--password <password>    password for login
-u,--user <user name>       user for login (required)
-url <ior url>              url to connect to
-v,--verbose                enables verbose output
```

The options have the following meaning:

Table 3.25. Options of *dumpusers*

Property	Description
<code>-e</code>	Defines the encoding for the output. Default is UTF-8. An encoding can only be used when the tool dumps in a file.
<code>-f</code>	File name of the user repository dump.

Encryptpasswords

Using the `cm encryptpasswords` utility will encrypt all passwords (to be more strict, the hash values of the passwords) stored in the database with a 256 bit key on basis of the AES algorithm (Rijndael). When starting the utility, make sure that the corresponding *CoreMedia Content Server* is not running.

Encrypting the passwords of a *Replication Live Server* needs slightly more care:

1. Set the property `replicator.enable` in the file `replicator.properties` to `false`.
2. Start the server.
3. Wait until the initial replication is complete.
4. Stop the server.
5. Encrypt the passwords with `cm encryptpasswords`.
6. Set the property `replicator.enable` in the file `replicator.properties` back to `true`.

The utility program is executed with:

```
cm encryptpasswords -encrypt
```

During operation, the utility writes some output to indicate the progress of encryption.

The generated key is written to the file `$INSTALL_DIR/etc/keys/<database name>.<dbuser>.rijndael`. Do not delete this key file and instead make sure that a backup exists in a safe place. Without the file, it is no longer possible to log in. You must copy this file to the Content Server installation under `WEB-INF/etc/keys` (The path can be configured by setting the property `cap.server.encryptpasswords.keyfile` in `contentserver.properties`). If you want to install a new server and you still want to use the old database the key file from the old installation must be present in the new installation. Likewise, if you want to install and use a new database you have to delete the key file. Otherwise, the program would try to decrypt the new decrypted passwords.

When the utility is used more than once, the passwords will be re-encrypted with a new key. No harm can occur.

If you want to revert to decrypted passwords, run the following command and remove the key file from the server installation afterwards:

```
cm encryptpasswords -decrypt
```

Groovy Shell

With the `groovysh` utility program, you can interactively access the Content Server, using *Unified API* commands. The Groovy Shell is a third-party tool maintained by the Groovy community. See <http://groovy-lang.org/groovysh.html> for details

Before you can start the *Groovy Shell*, copy the file `modules/cmd-tools/cms-tools-application/target/cms-tools/bin/groovysh.profile` into your home directory as `.groovy/groovysh.profile`. This file defines the login parameter (username, password, IOR URL for the *Content Server* and creates connections to the *Unified API* repositories. Adjust the `iroUrl` property to the corresponding URL of the *Content Server*. Have a look into the profiles file in order to learn how to address the repositories.

You can either start the Groovy Shell as a command line client (default) or the Groovy Console which offers a GUI. In order to start the *Groovy Console*, open the `groovysh.jpif` file and assign the `groovy.ui.Console` class to the `JAVA_MAIN_CLASS` property. The *Groovy Console* does not load the content from the profile file. So, copy the content of the profile file into the console and execute the script to set the required variables.

The program is executed with:

```
modules\cmd-tools\cms-tools-application\target\cms-tools\bin cm groovysh
```

The shell is now active or the console opens and you can enter your UAPI commands. Find the UAPI JavaDoc at <http://documentation.coremedia.com/dxp8/> following the CMS Javadoc link.

Get, for example, the user named "Rick":

```
ur.getUserByName('Rick')
```

```

Groovy Shell (2.2.1, JVM: 1.8.0_51)
Type 'help' or '\h' for help.
-----
groovy:000> ur.getUserByName('Rick')
==> User [coremedia:///cap/user/14]
groovy:000>

```

Configuration

Using the Groovy Console

Figure 3.1. Result of a command in Groovy Shell

JMXDump

The `jmxdump` utility prints out all JMX parameters and its values of the specified application server.

```
usage: cm jmxdump -u <jmx url> [other options] [
  -b <objectname#attribute>|
  -c <credential>|
  -v]

Available options:

-b,--mbean <arg>           MBean name and attributes:
                             name#attribute,...,attribute. Might
                             contain placeholder '*'
-c,--credential <arg>     Credentials when connecting to the
                             remote mbean server
-u,--url                    JMX service url
-v                           More verbosity
```

Usage of `jmxdump`

The options have the following meaning:

Parameter	Description
-b <objectname#attribute>	The MBean name and attributes: name#attribute,...,attribute. Might contain placeholder '*'
-c <arg>	Credentials when connecting to the remote mbean server
-u <url>	JMX service url. E.g.: service:jmx:rmi://<server url>:<JMX server port>/jndi/rmi://<server url>:<JMX registry>/jmxrmi
-v	More verbosity

Table 3.26. Options of `jmxdump`

The default ports for the URL parameter can be found in the section "Port Reference" of the Project Developer Guide.

Killsession

This utility kills specified sessions on the *Content Server*. You can obtain the IDs of the open sessions with the `sessions` utility (see [Section "Sessions" \[103\]](#)).

Well-behaved clients terminate their sessions, and the content server automatically kills sessions which don't respond within a certain timeout. So this is an emergency tool which you will need in exceptional cases only.

```
usage: cm killsession -u <user> [other options] <session> ...
available options:
-d,--domain <domain name>  domain for login (default=<builtin>)
-p,--password <password>  password for login
```

Usage of `killsession`

```
-u,--user <user name>      user for login (required)
-url <ior url>             url to connect to
-v,--verbose                enables verbose output
```

That is, `killsession` has none but the standard options.

Restoreusers

The `restoreusers` tool reads an XML file which has been written using the `dumpusers` tool. The structure of the XML file is defined by the `lib/xml/core/media-userrepository.xsd` schema. It contains a nested structure of group elements which contain rules defined for this group and elements representing the members of this group. To provide for membership in multiple groups a `userref` or `groupref` element can be used which refers back to a previously defined user or group.

Members are identified by their `capid` attribute, if given. If no `capid` is given, the member is identified by name and domain. If no such member is found, a new member is created. Members can only be created in the built-in user repository!

The identified member is updated to the corresponding values in the XML file, such as name, password, home folder and a group's `isAdministrative` flag. A new home folder is created if the given path does not exist yet.

Many attributes cannot be changed once a group or user has been created (the domain of a user or group, the name or password of an external (LDAP) user, the flags `isContentGroup` and `isLiveGroup` of a group, the name and the members of an external group). If a mismatch is detected, the `restoreusers` tool exits with an error message. In addition, the tool can only add or change rules (including rules on external groups) and can only add memberships, but cannot remove them.



When a rights rule refers to a non-existent content path, an empty folder will be created at the indicated location, unless the rule's `createFolder` attribute is set to false. If `createFolder` is false and the folder does not exist, the rule is ignored on import.

```
usage: cm restoreusers -u <user> [other options]
       -f <dump-file-url>
```

available options:

```
-d,--domain <domain name>  domain for login (default=<builtin>)
-f,--file <file>           url of the user repository dump
-p,--password <password>  password for login
-u,--user <user name>     user for login (required)
```

Example 3.22. Usage of the `restoreusers` tool

<code>-url <ior url></code>	url to connect to
<code>-v,--verbose</code>	enables verbose output

The options have the following meaning:

Parameters	Description
<code>-f</code>	The file name of the user repository XML file.

Table 3.27. Parameters of restoreusers

Runlevel

With the `runlevel` tool you control the mode of operation of the content server. See [Section 2.4, “Server Run Levels” \[25\]](#) for details about runlevels. You should always use `runlevel` to shut down the server.

```
usage: cm runlevel -u <user> [other options] -r <runlevel>
      [-g <grace>] [-w <timeout>]
      runlevel -u <user> [other options] -a
      runlevel -u <user> [other options] -a
available options:
-g,--grace <grace>      grace period in seconds
                        (default=120),
                        use jointly with the option -r
-w,--wait <wait>       wait time in seconds (default is
                        not to wait for runlevel change),
                        use jointly with the option -r
-a,--abort              abort pending runlevel switch
-d,--domain <domain name> domain for login (default=<builtin>)
-p,--password <password> password for login
-r,--runlevel <runlevel> desired runlevel (one of offline,
                        maintenance, administration, online)
-u,--user <user name>  user for login (required)
-url <ior url>          url to connect to
-v,--verbose            enables verbose output
```

Usage of runlevel

The options have the following meaning:

Parameter	Description
<code>-g</code>	Delays the runlevel switch for the specified number of seconds. This gives users the chance to save their changes and logout. During the grace period the runlevel switch can be aborted with the <code>-a</code> option.
<code>-w</code>	If used with <code>-r</code> , the utility will not exit before the target runlevel has been reached (exit code 0) or the specified number of seconds have passed (exit code 1). The utility will not fail, if a runlevel change has already been scheduled (it will not reschedule another change, but just wait for the specified runlevel to be reached). If the server is not available when the utility is run,

Table 3.28. Options of runlevel

Parameter	Description
	it will not fail but keep trying to connect or time out after the specified number of seconds. If time out is met, <code>runlevel</code> will exit with exit code 1.
<code>-a</code>	Abort a pending runlevel switch triggered by a recent invocation of <code>runlevel</code> .
<code>-r</code>	Specify the new runlevel. The legal values are described in the [Operations Basics Manual]

Note that you cannot switch the runlevel from `offline` mode. You have to restart the server instead.

Schemaaccess

You can use the `schemaaccess` tool to perform database actions on the user's database schema.

`schemaaccess` uses SQL to work directly on the database. Only use this tool when you are familiar with the database structure of CoreMedia applications.



Example 3.23. Usage of `schemaaccess`

```
Usage: SchemaAccess <action> [-p|-actionParameters <parameters>]
(to use sql.properties settings)
or
SchemaAccess <driver> <jdbc:url> <user> <password> <dbtype>
<action> [-p|-actionParameters <parameters>]

Available Options:
-p|-actionParameters <parameters>: Parameters for the action

Choose <action> from:
  showTables
  dropTables
  showViews
  dropViews
  showSequences
  dropSequences
  showIndices
  showAll
  dropAll
  updateStatistics
  clearTables
```

As is shown above, you can either add the database connection parameters to the call of `schemaaccess` or only give the action as parameter and use the settings from `<CoremHome>/properties/corem/sql.properties`.

The actions have the following meaning:

Table 3.29. Schemaaccess actions

Action	Description
showTables	Shows all table names of the schema owner.
dropTables	Drops all the user's tables in the database schema. This does not delete blobs that are stored on the hard disk!
showViews	Shows all views of the schema owner.
dropViews	Drops all the user's views in the database schema.
showSequences	Shows all sequences of the schema owner.
dropSequences	Drops all the user's sequences in the database schema.
showIndices	Shows all indices of the schema owner.
showAll	This action executes <i>showTables</i> , <i>showViews</i> , <i>showSequences</i> and <i>showIndices</i> in one call.
dropAll	This action executes <i>dropTables</i> , <i>dropViews</i> and <i>dropSequences</i> in one call.
updateStatistics	Updates the statistics of the user's tables and indices in the database schema.
clearTables	Deletes all data from the tables given as parameters. If no tables are given, then all tables from the schema are cleared.

Serverimport/Serverexport

CoreMedia content items can be exported into XML files in the file system with the `cm serverexport` utility. These files can be imported again with `cm serverimport`. The `cm serverimport` is different from the *CoreMedia XML Importer*, which is described in the [Importer Manual].

The following limitations for export exist:

- Only the latest version of a content item is exported.
- No metadata, such as the last modification date or the status 'published', is exported.
- For content items marked for deletion, only name and path are exported.
- Empty folders are not exported, because they contain no content items.

From the limitations mentioned above the following consequences arise for import:

- Only the latest version of a content item can be imported.
- content items which were marked for deletion are empty, that is all property fields contain null values, such as empty strings, zeros, etc.

- Every imported content item has the status "checked in" and "not approved".
- Content items with invalid XML property values will not be imported. You can force an import of such content items with the option `--no-validate-xml` (not recommended).
- Content items with link list property values that violate the minimum or maximum cardinality for the property type will not be imported. You can force an import of such content items with the option `--no-validate-link-cardinality` (not recommended).

XML files are created so that internal links to these content items are maintained after import. On import of the complete content of a *CoreMedia Server*, the content, the folder structure of the content items and the linking of the content items is maintained for content items not marked for deletion.

The properties of the content types on the export server do not have to match those on the import server. Those which are not available on the import server are ignored, and those which exist in addition are filled with null values.

If a content item already exists on the import server, then a new version of the content item will be created.

Multi-Site

Localizable content items have two special properties `master` and `masterVersion`, both defined in the `SiteModel`. Because versions are lost on export / import, `serverimport` and `serverexport` have a special handling built in for these two properties in order to set a reasonable `masterVersion` during an import. For details consult Section "ServerImport and ServerExport" in *CoreMedia Digital Experience Platform 8 Developer Manual*.

Server Import

The Importer command has the following syntax:

cm serverimport -u <user> [other options] (<file> | <dir>)...

The options have the following meaning:

Parameter	Description
<code>-r, --recursive</code>	Recursive import of files and subdirectories on entry of a directory to be imported
<code>-h, --halt</code>	Halt on error
<code>-v, --verbose</code>	enables verbose output

Table 3.30. Parameters of the `serverimport` utility

Parameter	Description
<code>--no-validate-xml</code>	Disables XML property value validation. By default, content items with invalid XML property values will not be imported. This option can be used to force an import of such documents (not recommended).
<code>--no-validate-link-cardinality</code>	Disables validation of link list cardinalities. By default, documents with link list property values that violate the minimum or maximum cardinality for the property type will not be imported. This option can be used to force an import of such documents (not recommended).
<code>--skip-entities</code>	Skips resolution of external XML entities in imported files. By default, XML entities are resolved which may trigger requests to external servers.
<code>-t, --threads <threads></code>	Use the given number of threads for importing content. Multiple threads may increase throughput in the presence of network and database latency, and may increase CPU utilization. Default: 1
<code>-u, --user <user name></code>	Name of the user.
<code>-d, --domain <domain></code>	The domain of the user.
<code>-p, --password <password></code>	Password of the user. The tool will prompt the user for a password if not specified as option.
<code>-url <ior url></code>	The IOR URL of the Content Server.

On executing the program, the path of at least one XML file or directory must be given as argument. Relative paths are allowed and refer to the current directory in which the program was started.

Server Export

The exporter command has the following syntax:

cm serverexport -u <user> -b <basedir> [other options] [<id> | <path>]...

The options have the following meaning:

Parameter	Description
<code>-r, --recursive</code>	Recursive export of files and subdirectories on entry of a directory for export.

Table 3.31. Parameters of the `serverexport` utility

Parameter	Description										
<code>-b, --basedir <basedir></code>	The directory in which the exported XML files are saved. A relative path is relative to the working directory.										
<code>-v, --verbose</code>	enables verbose output										
<code>-enc</code>	The encoding of the server export. "ISO-8859-1" creates a server export in Iso-Latin-1. The default value is "UTF-8" which creates an export in Unicode. Used values for coding must be supported by Java.										
<code>-cut <length></code>	The <code>serverexport</code> utility exports the files in a directory structure which reflects the folder structure of the CoreMedia repository. The resulting path length may exceed OS limits, so you can use the <code>-cut</code> option to limit the maximum path length. By default, no limit will be applied. The shortened path will not affect the reimport of the exported files.										
<code>-pretty</code>	Specifies if the exported XML files should be pretty printed.										
<code>-l, --lint <warning></code>	<p>Enables different warnings to be logged. The following warnings can be specified:</p> <table border="0"> <tr> <td><i>linkignored (default)</i></td> <td>Warn on links to either destroyed or deleted contents which are ignored on export.</td> </tr> <tr> <td><i>linkmissing</i></td> <td>Warn on links whose target is not part of the export. This might lead to inconsistencies on import: link target might not be available or link target's content might be outdated.</td> </tr> <tr> <td><i>translationstate</i></td> <td>Warn on translation states (see Multi-Site) which are considered harmful especially because they cannot be rebuilt exactly on import.</td> </tr> <tr> <td><i>all</i></td> <td>Warn on all detected issues.</td> </tr> <tr> <td><i>none</i></td> <td>Disable warnings.</td> </tr> </table>	<i>linkignored (default)</i>	Warn on links to either destroyed or deleted contents which are ignored on export.	<i>linkmissing</i>	Warn on links whose target is not part of the export. This might lead to inconsistencies on import: link target might not be available or link target's content might be outdated.	<i>translationstate</i>	Warn on translation states (see Multi-Site) which are considered harmful especially because they cannot be rebuilt exactly on import.	<i>all</i>	Warn on all detected issues.	<i>none</i>	Disable warnings.
<i>linkignored (default)</i>	Warn on links to either destroyed or deleted contents which are ignored on export.										
<i>linkmissing</i>	Warn on links whose target is not part of the export. This might lead to inconsistencies on import: link target might not be available or link target's content might be outdated.										
<i>translationstate</i>	Warn on translation states (see Multi-Site) which are considered harmful especially because they cannot be rebuilt exactly on import.										
<i>all</i>	Warn on all detected issues.										
<i>none</i>	Disable warnings.										
<code>-fow, --fail-on-warning</code>	Fail if any of the warnings configured by <code>lint</code> occurs										
<code>-fae, --fail-at-end</code>	Only fail at end providing a summary of all issues found.										

Parameter	Description
<code>-u, --user <user name></code>	Name of the user.
<code>-d, --domain <domain></code>	The domain of the user.
<code>-p, --password <password></code>	Password of the user. The tool will prompt the user for a password if not specified as option.
<code>-url <ior url></code>	The IOR URL of the Content Server.
<code>--blobsizelimit <size></code>	Used in conjunction with <code>-s</code> . Default: 1MB. Blobs larger than the given number of bytes are stored in the directory defined by <code>sharedblobbasedir</code> .
<code>-s, --sharedblobbasedir <sharedblobbasedir></code>	The directory in which to store blobs that are larger than the size given by <code>blobsizelimit</code> . Equal blobs will be stored only once.

As optional arguments, the IDs or the folder paths of CoreMedia resources can be entered.

Example:

```
cm serverexport -r -u admin -p admin -b /export 7531
```

This call of `serverexport` will export all content items and subfolders (`-r`) of the folder with the ID 7531 into the `/export` directory. The program logs in at the server as the admin user, using the admin password (which should never be admin as in the example).

SQL

The program `cm sql` is used to access the databases of Content Servers manually. This program should only be used by those with precise knowledge of the SQL query language as well as of the table structure of the CoreMedia system. Table contents can be displayed or manipulated with it.

Overview

Only use read commands on the database when the *CoreMedia Content Server* is running. If you want to write data via **cm sql**, be sure that the Content Server is down. Otherwise, data corruption can occur.



The usage is:

Usage

```
cm sql [-script <scriptname>]
```

If a SQL script is passed via the option `-script`, the script will be executed.

After entering `cm sql`, a connection to the CoreMedia system database is opened using the database settings configured in `sql.properties` (see [Section 5.3, "Configuration in `sql.properties`" \[240\]](#)).

If the command is carried out in a Windows environment, a graphical user interface opens which allows SQL commands to be entered.

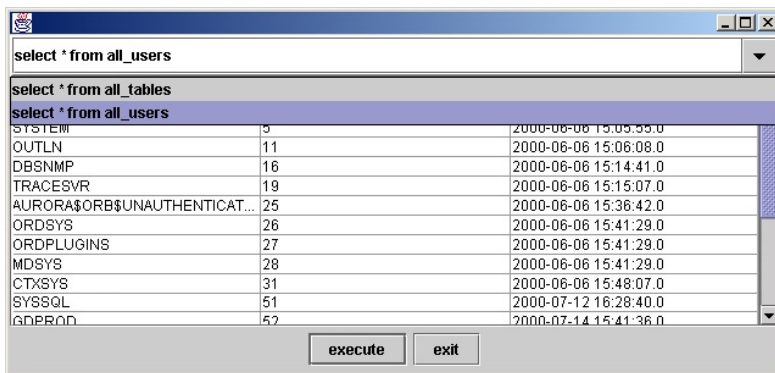


Figure 3.2. CM SQL user interface

If there is no Windows environment active (Unix: no `DISPLAY` variable set), a command line prompt appears which allows SQL commands to be entered.

```
DBConnection: opened to: jdbc:oracle:thin:@server:1521:DEVELOP
for user: CMPROD
database: Oracle
version: Oracle8i Enterprise Edition Release 8.1.6.0.0 -
Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production
driver: Oracle JDBC driver
version: 8.0.5.2.0

Enter your SQL-Statement, finish with ';'
SQL>
```

Example 3.24. CM sql command line operation

Keep in mind that you have to terminate a SQL statement on the command line with a semicolon but in the graphical interface a semicolon is not allowed as a terminator of the statement.



Tracesession

This utility can be used to start and stop tracing (logging) of specified sessions on the content server. The logging must be configured with the log facility `trace`. All sessions traced by `tracesession` will log in the same file. See the [Operations Basics Manual] for details on the logging of CoreMedia CMS.



Use this utility with care! `tracesession` will produce additional load on the server and will slow down the system. Only use `tracesession` for debugging custom clients or to determine the cause for unusual peaks in the server load but not for daily logging. The output of `tracesession` is very technical, so you probably will only benefit from it in cooperation with a CoreMedia consultant or the CoreMedia support.

```
usage: cm tracesession -u <user> [other options] -s <session> [-q]
      tracesession -u <user> [other options] -t <type> [-q]
available options:
-d,--domain <domain name>    domain for login (default=<builtin>)
-p,--password <password>    password for login
-q                             quit tracing
-s,--session <session>      ID of the session to be traced
-t,--type <type>            type of the sessions to be traced
                             (one of: unknown, editor,
                             generator, publisher, replicator,
                             watchdog, importer, utility, uapi,
                             all)
-u,--user <user name>       user for login (required)
-url <ior url>              url to connect to
-v,--verbose                 enables verbose output
```

Usage of tracesession

The options have the following meaning:

Parameter	Description
-q	Stop tracing the specified sessions
-s	Specify a session by its ID. You can obtain the IDs of all sessions with the <code>sessions</code> tool (see Section “Sessions” [103]).
-t	Specify the sessions to be traced by their common type. Run the <code>sessions</code> tool with the <code>-v</code> option to find out about the types of the open sessions. The legal values for this parameter (see usage) are obvious shortcuts for those types. There are two special values, <code>unknown</code> and <code>all</code> . <code>all</code> denotes all sessions, <code>unknown</code> denotes all sessions but those of the concrete type values.

Table 3.32. Options of tracesession

CoreMedia recommends that you separate the `tracesession` logging from the standard logging (which usually goes to `capserver.log`), because a mixture of both looks pretty cluttered and is hard to read. Just configure an additional log action in `contentserver.properties`, for example


```
# write all trace output to a log file
log.action.2.class=FileAction
log.action.2.selectors=trace:debug
log.action.2.initArgs=\
file=var/logs/capservers-trace.log, fileSize=4000, writeOps=1000
```

Now you can start `tracesession`, for example a user's editor session which you found with the `sessions` tool:

```
> cm tracesession -u admin -p admin -s coremedia:///cap/session/14
```

Traces started by ID and by type are independent of each other. That is, you cannot stop tracing the above session with the following command, although it is an editor session:

```
> cm tracesession -u admin -p admin -t editor -q
```

You can only stop it by ID, just like you started it.

Unlockcontentserver

The `unlockcontentserver` tool recovers a database schema after a server aborted without cleaning up the database. In this case it can happen that a persistent lock remains in the database. This lock must be cleared explicitly, because the server would otherwise suspect that a second server is running on the same database.

This utility has the following syntax:

```
cm unlockcontentserver
```

You have to run the tool in the server installation where it can access the `sql.properties` configuration file.

Usedlicenses

CoreMedia CMS uses a license system based - among others - on named users (see the [Operations Basics Manual] for details). If all named user licenses are consumed, no other user can login. So you might want to free a license for a new user.

`usedlicenses` is the utility for this use case. You can use it to view all used named licenses and to remove named licenses from a user. Keep in mind, that a user can be registered with more than one service. Used licenses are shown as a table with the column names `username` and `servicename`:

- `username` holds user IDs,
- `servicename` one out of the following CoreMedia service names:

debug, dotnetappbridge, feeder, editor, filesystem, importer, publisher, replicator, system, webserver, workflow.

You can find the ID of a user in the *User Management Window* of the *Site Manager* or with the `dumpusers` tool.

If you use the tool on a given user, all licenses for all services occupied by this user are removed from the user. Nevertheless, the user will not be kicked out of a current editor session and the license will not be usable for a new user. You have to restart the *Content Server* first.

```
Usage: cm usedlicenses (-print|-deleteuser <user>)
```

You can use either the user's id or the user's name. In the rare case that the user's name consists entirely of numerals, you must use the user's id.

The parameters have the following meaning:

Parameter	Description
<code>-print</code>	Prints out the named licenses in use
<code>-deleteuser <user></code>	Deletes all licenses (thus making them available for reuse) for the user indicated by either the user id or the user name

Table 3.33. Parameters of the `usedlicenses` utility

Example

A new user tries to log in the *Site Manager* but gets a "licenses exhausted" message. So, you want to free a license for the user.

1. Show all used licenses with `cm usedlicenses -print`
2. Select a user from whom you want to revoke the license. For the sake of the example assume that the user has the ID 8.
3. Delete the license with `cm usedlicenses -deleteuser 8`

If you view the used licenses again, you will see that the license has been removed. Nevertheless, you have to restart the *Content Server*.

4. Make sure that all users have saved their current work, then restart the server.

Now you are done, a new user can log in.

3.14.3 Repository

The Repository tools described in this section perform some basic editorial tasks.

Repository tools

- `approve`: Approves a resource.
- `bulkpublish`: Publish or withdraw all resources below a given folder.
- `destroy`: Delete resources from the repository.
- `publish`: Publishes a resource.
- `publishall`: Publish all resources to a newly set up *Master Live Server*, initializing or recreating the live repository.
- `republish`: Republish a set of resources which have been published in the recent past.
- `query`: Execute a structured query in the repository.
- `queryapprove`: Execute a structured query in the repository and approve the resulting content items.
- `querypublish`: Execute a structured query in the repository and publish the resulting content items.
- `search`: Execute a full-text search in the repository.

Approve

With the `approve` tool you can approve resources.

Usage of approve

```
usage: cm approve -u <user> <other options> [ -l <limit> ]
      [ <idl>... | -I <idl>... | -t <pathl>... |
        -cq <queryl>... ]

available options:

-c,--contents <contents>      ids of contents to place approve.
                                Ids are represented as
                                'coremedia:///cap/content/1234'
-cq,--contentquery <query>    query/queries to select contents
-d,--domain <domain name>     domain for login (default=<builtin>)
-I,--id <id>                  id(s) to select content objects
-l,--limit <limit>           limits the number of content objects
                                to select, negative for unlimited
                                which is also the default
-p,--password <password>     password for login
-t,--path <path>             path(s) to select content objects
-u,--user <user name>        user for login (required)
-url <ior url>               url to connect to
-v,--versions <versions>     ids of versions whose properties to
                                approve.
                                Ids are represented as
                                'coremedia:///cap/version/1234/56'
-V,--verbose                  enables verbose output
```

The options have the following meaning:

Table 3.34. Options of approve

Parameter	Description
<code>-c</code>	Ids of resources to be approved. Ids have the following format: <code>coremeda:///cap/content/1234</code> .
<code>-v</code>	Ids of content item versions to be approved. Ids have the following format: <code>coremeda:///cap/version/1234/56</code> . 56 must be replaced by the version number you want to approve.
<code>-cq <query></code>	Content Selection Content Query to locate contents. Parameter can be used multiple times. Results will be ORed.
<code>-I <id></code>	Content Selection Select content objects via explicit id. This is an alternative if you want to mix explicit ids with for example content queries. Parameter can be used multiple times. Results will be ORed.
<code>-t <path></code>	Content Selection The path of a resource to be approved. Parameter can be used multiple times. Results will be ORed.
<code><ids></code>	Content Selection IDs to be approved. If mixing with other content selection parameters it is recommended to explicitly use the parameter <code>-I</code> Parameter can be used multiple times. Results will be ORed.
<code>-l <limit></code>	Content Selection Limit the number of selected contents. Default is unlimited.

For information about the UAPI query syntax see Section 5.5, “Query Service” in *CoreMedia Unified API Developer Manual*.



If no options are used or if you specify the resources via content selection parameters, `approve` approves the resources eagerly:

- ➔ If the resource is checked out, `approve` checks it in first.
- ➔ The latest version is approved.
- ➔ The place is approved.

If you set the `-c` flag, only the places of the resources are approved. If you set the `-v` flag, only the specified versions are approved.

Bulkpublish

With the `bulkpublish` tool you can publish or withdraw all resources below a given folder. If necessary, you can automatically check-in resources and approve them.

Example 3.25. Usage of `bulkpublish`

```
usage: cm bulkpublish -u <user> [other options] [-f <path>]
available options:
-a,--approve                combines -ap and -av
-ap,--approve-places        approve if not place approved
-av,--approve-versions      approve the latest checked-in
                             version
-b,--publish                publish if approved, but don't
                             delete or withdraw
-ub,--unpublish             withdraw unpublish if approved,
                             but don't delete
-bs,--batchsize <arg>      number of approve/placeApprove
                             operations to execute per batch
                             (default 1000)
-c,--checkin                checkin checked out contents
-d,--domain <domain name>  domain for login
                             (default=<builtin>)
-f,--folder <folder>       base folder for all operations
                             (mandatory only for multisite
                             systems)
-p,--password <password>   password for login
-u,--user <user name>       user for login (required)
-url <ior url>              url to connect to
-v,--verbose                enables verbose output
```

The options have the following meaning:

Parameter	Description
-a	Equivalent to the combination of <code>-ap</code> and <code>-av</code> .
-ap	Approve the place of any content below the base folder that is not place approved yet.
-av	Approve the latest checked-in version of each document below the base folder.
-b	Publish all resources below the base folder that are approved but not published yet and that are not marked for deletion or withdrawal.
-ub	All resources below the base folder that are published will be withdrawn from the <i>Master Live Server</i> .
-c	Check-in all content items below the base folder which are checked-out.
-f	The path of a folder for which you want to start the operations (such as <code>/articles/sport</code>). If no path is entered, all resources below the root folder will be used. If you use a multi-site enabled system, it is mandatory to enter a path to a folder that belongs to a single base folder (see Section 2.3, "Multi-Site Publishing" [23]). This option can be given multiple times.

Table 3.35. Options of the `bulkpublish` tool

Destroy

With the `destroy` tool you can delete resources from the repository.

By means of the `destroy` tool, you remove the indicated objects permanently and irrevocably. Use with great care. This tool is typically needed to recover from error conditions only.



Usage of `destroy`

```
usage: cm destroy -u <user> <other options> [ -f ] [ -R ]
      [ -l <limit> ] [ <id1>... | -I <id1>... |
      -t <path1>... | -cq <query1>... |
      -vq <query1>... ]

available options:

-cq,--contentquery <query>  query/queries to select contents
-d,--domain <domain name>  domain for login (default=<builtin>)
-f,--force                  force destruction of folders with
                             children
-I,--id <id>               id(s) to select content objects
-l,--limit <limit>        limits the number of content objects
                             to select, negative for unlimited
                             which is also the default
-p,--password <password>  password for login
-R,--recyclebin            include contents from recycle bin
-t,--path <path>         path(s) to select content objects
-u,--user <user name>     user for login (required)
-url <ior url>            url to connect to
-v,--verbose              enables verbose output
-vq,--versionquery <query> query/queries to select versions
```

The options have the following meaning:

Table 3.36. Options of `destroy`

Parameter	Description
<code>-f</code>	Force the destruction of non-empty folders, which is normally forbidden due to safety considerations
<code>-cq <query></code>	Object Selection Content Query to locate contents. For versions see <code>-vq</code> . Parameter can be used multiple times. Results will be ORed.
<code>-vq <query></code>	Object Selection Version Query to locate versions. For contents see <code>-cq</code> . Parameter can be used multiple times. Results will be ORed.
<code>-R</code>	Object Selection Add contents from recyclebin. Could also be done via <code>-cq</code> .
<code>-I <id></code>	Object Selection Select objects via explicit id. This is an alternative if you want to mix explicit ids with for example content queries. Parameter can be used multiple times. Results will be ORed.

Parameter	Description
<code>-t <path></code>	Object Selection Path of a resource to be destroyed. Parameter can be used multiple times. Results will be ORed.
<code><ids></code>	Object Selection IDs of objects to be destroyed. If mixing with other content selection parameters it is recommended to explicitly use the parameter <code>-I</code> Parameter can be used multiple times. Results will be ORed.
<code>-l <limit></code>	Object Selection Limit the number of selected objects to the given value. Default is unlimited.

For information about the UAPI query syntax see Section 5.5, “Query Service” in *CoreMedia Unified API Developer Manual*.



In order to keep *Content Management Server* and *Master Live Server* in sync, `destroy` tries to withdraw the resources from the *Master Live Server* before it destroys them on a *Content Management Server*. To avoid dead links on the *Master Live Server*, all referencing content items are also withdrawn. On the *Content Management Server* however, the referencing content items are not destroyed, and thus dead links may arise.

This tool might affect your live site.



As stated above, the `destroy` tool tries to withdraw the content that should be deleted and all content that recursively links to this content! That is, if the content to be deleted is linked by other content, more content might be withdrawn from your live site than you expected.

So when you use the tool, be sure that only content you want to remove from the live site links to the content you want to destroy and keep in mind that `destroy` is an emergency tool.

`destroy` is an emergency tool. The normal way to get rid of resources without causing dead links is to move them to trash and have them deleted by the cleaners (see Section “Cleaners” [118]).

Concerning users and groups, `destroy` is not an LDAP tool. If you apply `destroy` to LDAP users or groups they will soon reappear in the *CoreMedia CMS* if the LDAP server and the corresponding `UserProvider` still serve them. If you delete users and groups on the LDAP server, the *CoreMedia CMS* notices this automatically. So you don't need `destroy` in the regular case but only to fix inconsistencies caused by errors or misconfiguration of user providers.

Publish

With the `publish` tool you can publish resources.

```
usage: cm publish -u <user> <other options> [ -l <limit> ]
      [ <idl>... | -I <idl>... | -t <path>... |
        -cq <query>... ]

available options:

-cq,--contentquery <query>    query/queries to select contents
-d,--domain <domain name>    domain for login (default=<builtin>)
-I,--id <id>                  id(s) to select content objects
-l,--limit <limit>           limits the number of content objects
                              to select, negative for unlimited
                              which is also the default
-p,--password <password>    password for login
-t,--path <path>            path(s) to select content objects
-u,--user <user name>       user for login (required)
-url <ior url>              url to connect to
-v,--verbose                 enables verbose output
```

Usage of publish

The options have the following meaning:

Table 3.37. Options of publish

Parameter	Description
<code>-cq <query></code>	Content Selection Content Query to locate contents. Parameter can be used multiple times. Results will be ORed.
<code>-I <id></code>	Content Selection Select content objects via explicit id. This is an alternative if you want to mix explicit ids with for example content queries. Parameter can be used multiple times. Results will be ORed.
<code>-t <path></code>	Content Selection The path of a resource to be published. Parameter can be used multiple times. Results will be ORed.
<code><ids></code>	Content Selection IDs to be published. If mixing with other content selection parameters it is recommended to explicitly use the parameter <code>-I</code> Parameter can be used multiple times. Results will be ORed.
<code>-l <limit></code>	Content Selection Limit the number of selected contents. Default is unlimited.

For information about the UAPI query syntax see Section 5.5, “Query Service” in *CoreMedia Unified API Developer Manual*.



The `publish` tool publishes the very latest versions (including working revisions of checked out content items) of the specified resources immediately. This includes all mandatory preliminary actions:

- ➔ Checking in the resource
- ➔ Approving the latest version
- ➔ Approving the place (for the whole path)

`publish` does not cause dead links on the master server but also publishes all referenced resources if necessary. However, those referenced resources are not published as eagerly as the specified resources. If a referenced resource is already published, nothing is done. Otherwise, the latest approved version is published. If there is no approved version, the very latest version is published.

Publishall

The `publishall` tool is used in two cases: One, to quickly initialize a Master Live Server with the current contents of the Content Management Server, usually in a quality assurance or continuous integration environment; and two, to populate a new Master Live Server in case the original *Master Live Server* database has been corrupted and is not recoverable from a consistent backup. Before starting the `publishall` tool, you must create an empty database schema, configure it in the *Master Live Server*, (re)start the server and wait until it is completely initialized. You can then publish all content that is marked as published to the new server as follows:

```
cm publishall [ -a [ -cq <query> ] ] [ -t <threads> ]
  <cmsiorurl> <cmsuser> <cmspwd>
  <masteriorurl> <masteruser> <masterpwd>
```

Example 3.26. Usage of `publishall`

The parameters have the following meanings:

Parameter	Description
<code>-a</code>	If given, publish all contents on the CMS (by default, except /Home). Otherwise, only publish contents already marked as published (that is, which were previously published to a lost or damaged MLS).
<code>-cq <query></code>	If given together with <code>-a</code> , publish all contents matching the given query. Note that there must not be any links from published contents to non-published contents, which for performance reasons is not checked by the <code>publishall</code> tool. default: "NOT BELOW PATH '/Home'"
<code>-t <threads></code>	Use the given number of threads for creating content on the master live server. Multiple threads may increase throughput in the

Table 3.38. Parameters of the `publishall` utility

Parameter	Description
	presence of network and database latency. Default: 1
<code>cmsiorurl</code>	The IOR URL of the <i>Content Management Server</i>
<code>cmsuser</code>	The user on the <i>Content Management Server</i> to be used for reading (typically admin)
<code>cmspwd</code>	The password on the <i>Content Management Server</i>
<code>masteriorurl</code>	The IOR URL of the <i>Master Live Server</i>
<code>masteruser</code>	The user on the <i>Master Live Server</i> to be used for writing (typically admin)
<code>masterpwd</code>	The password on the <i>Master Live Server</i>

When there are rights rules for live groups to be published, required groups of the built-in user management will be automatically created, but the subgroup relationships must be established separately. This is in line with the normal behavior of the publisher.

During operation of the `cm publishall` tool, no changes should be made on the *Content Management Server*.

Even in a multi-site environment, the tool publishes all content to the *Master Live Server*. Afterwards, you may destroy unneeded parts of the repository with `cm multisitecleanup` as described in [Section “Splitting Content to Multiple Targets” \[72\]](#).

Replication Live Servers that have been connected to the replaced *Master Live Server* cannot be connected to the new *Master Live Server*. They must be rebuilt, too.

The `cm publishall` tool will reset the publication date and the publisher user of all contents in the content management environment, and the creation date of all contents in the live environment. The original information about publication dates is irretrievably lost when the `cm publishall` tool is used. Your business logic should therefore not rely too much on such metadata.

Republish

You can use the `republsh` tool to publish all content and versions below a given folder which have been published in a given period. A major use case is the update of a *Master Live Server* which has been recovered with a backup (see [Section 3.9.3, “Recovery of a Master Live Server Database” \[51\]](#)).

There are certain limitations when using this tool:

- Because the tool spawns only a single publication, it should not be used when there are many thousands of resources to be published. In particular, for the recreation of a fresh *Master Live Server* from a *Content Management Server* the tool `cm publishall` described in [Section “Publishall” \[149\]](#) is more appropriate.
- The tool cannot repeat the publication of withdrawals and deletions.
- The tool cannot repeat the publication of resources that have been marked for withdrawal or deletion after being published.
- The tool cannot repeat the publication of moves and renames of resources that have been moved or renamed again after being published.
- If the publication of a new folder fails for any reason, no resources contained in the folder can be published.
- If the publication of a new content item fails or any reason, no content items linking to that content item can be published.

Example 3.27.

```
usage: cm republish -u <user> [other options] [-f <path>]
available options:
  -a,--after           Minimum date for documents to be
                        republished <default=2 days before
                        now>.
                        Format is yyyyMMddHHmmss
  -b,--before         Maximum date for documents to be
                        republished <default=now>.
                        Format is yyyyMMddHHmmss
  -d,--domain <domain name> domain for login (default=<builtin>)
  -f,--folder <folder> base folder for republication (mandatory
                        only for multisite systems)
  -p,--password <password> password for login
  -u,--user <user name> user for login (required)
  -url <ior url>      url to connect to
  -v,--verbose        enables verbose output
```

The options have the following meaning:

Table 3.39. Parameters of the republish utility

Parameter	Description
-a	Start date for content items to be republished. The format is <code>yyyyMMddHHmmss</code> . By default, the start date is two days before now.
-b	The end date for content items to be republished. The format is <code>yyyyMMddHHmmss</code> . By default, the end date is the current date.

Parameter	Description
<code>-f</code>	The base folder for republication. It's only mandatory for multi-site systems. By default, the root folder will be taken as the base folder.

Query

The query tool can be used to start a synchronous, structured query against the content repository. It's analogous to the query used within the editors but you can create more sophisticated queries with the query tool. You can also transform old queries from Query documents into the new syntax and execute them.

For more details about the query syntax and for more examples see Section 5.5, "Query Service" in *CoreMedia Unified API Developer Manual*.



```
usage: cm query -u <user> <other options> [ -xl <limit> ] [ -R ]
      [ -l <limit> ]
      [ <idl>... | -I <idl>... | -t <path1>... |
        -cq <query1>... | -vq <query1>... ]
```

available options:

```
-cq,--contentquery <query>    query/queries to select contents
-d,--domain <domain name>    domain for login (default=<builtin>)
-I,--id <id>                  id(s) to select content objects
-l,--limit <limit>           limits the number of content objects
                              to select, negative for unlimited
                              which is also the default
-m,--transform                transform a legacy query given by -q
                              into a UAPI query
-p,--password <password>     password for login
-q,--query <query>           query string; deprecated for
                              UAPI-queries, use contentquery or
                              versionquery instead
-R,--recyclebin               include contents from recycle bin
-t,--path <path>             path(s) to select content objects
-u,--user <user name>        user for login (required)
-url <ior url>                url to connect to
-v,--verbose                  enables verbose output
-vq,--versionquery <query>   query/queries to select versions
-x,--execute                  transform a legacy query into a UAPI
                              query and execute it, implies -m
-xl,--executelimit <limit>  limits the query results for
                              executed transformed legacy queries
                              (use -l for unlimited); by default
                              uses same value as denoted by '-l'
```

...

Example 3.28. Query usage

Table 3.40. Parameters of the query utility

Parameter	Description
<code>-cq <query></code>	Content Selection Content Query to locate contents. For versions see <code>-vq</code> . Parameter can be used multiple times. Results will be ORed.
<code>-vq <query></code>	Content Selection Version Query to locate contents. For contents see <code>-cq</code> . Parameter can be used multiple times. Results will be ORed.
<code>-R</code>	Content Selection Add contents from recyclebin. Could also be done via <code>-cq</code> .
<code>-I <id></code>	Content Selection Select content objects via explicit id. This is an alternative if you want to mix explicit ids with for example content queries. Parameter can be used multiple times. Results will be ORed.
<code>-q <query></code>	The query string. Especially to be used for transforming legacy queries at command line. For selection of contents the parameters <code>-cq</code> and <code>-vq</code> are recommended. Parameter can be used multiple times. Results will be ORed.
<code>-e</code>	If set, the query specified by <code>-q</code> searches for all versions of the content items. <i>Deprecated:</i> Use <code>-vq</code> instead.
<code>-xl <limit></code>	Limit the number of results for executing transformed legacy queries. If not given defaults to <code>-l</code> .
<code>-m</code>	With the <code>-m</code> option you can transform legacy queries (according to the <code>coremedia-query.dtd</code> DTD) into UAPI queries (according to the syntax below). You can specify the legacy query directly with the <code>-q</code> option or you can select via id, path or for example <code>-cq</code> contents of content type Query whose legacy query will then be transformed at command line (the Query document itself is left untouched). If you want to see the results of the query in addition to this you can add the parameter <code>-x</code> .
<code>-x</code>	Like <code>-m</code> , additionally executes the transformed query. Implies <code>-m</code> .
<code>-t <path></code>	Content Selection Paths to be added to the result or path to Query content items to be transformed and possibly executed. Parameter can be used multiple times. Results will be ORed.
<code><ids></code>	Content Selection IDs to be added to the result or id of Query content items to be transformed and possibly executed. If mixing with other content selection parameters it is recommended to explicitly use the parameter <code>-I</code> . Parameter can be used multiple times. Results will be ORed.
<code>-l <limit></code>	Content Selection Limit the number of selected contents to the given value. Default is unlimited.

The following example shows the output of a query which searches for all content items below the /MenuSite/Fish folder:

```
cm query -u admin -p admin -cq "BELOW PATH '/MenuSite/Fish'"
```

The result shows the ID, the type and the path/name of the found content items.

```
coremedia:///cap/content/210 <Picture>
/MenuSite/Fish/SoleCitrus_pic
coremedia:///cap/content/212 <Picture>
/MenuSite/Fish/SalmonCitrus_pic
coremedia:///cap/content/216 <Picture>
/MenuSite/Fish/FreshCod_pic
coremedia:///cap/content/218 <Dish>
/MenuSite/Fish/SoleCitrus
coremedia:///cap/content/222 <Dish>
/MenuSite/Fish/SpicyTrout
coremedia:///cap/content/716 <Dish>
/MenuSite/Fish/CarpDish
coremedia:///cap/content/718 <Dish>
/MenuSite/Fish/troutmeal
```

Example 3.29. Result of a query

The following example transforms the default query of user johndoe into a UAPI query:

```
cm query -u admin -p admin -m -t "/Home/johndoe/My Query"
```

Note, that the name of the default query content item is locale specific. For German editors it is "Meine Recherche". The result shows the transformed query without executing it.

```
Legacy query:
<QUERY VERSION="6" LIMIT="50"><DOCUMENTTYPES VALUE="Document ">
<AND><AND><BOOLEAN VALUE="true"/><ISDESCENDANTOF VALUE="311"/>
</AND><LATEST/></AND></DOCUMENTTYPES></QUERY>

UAPI query:
TYPE "Document ": (BELOW ID 'coremedia:///cap/content/311') AND
((version=workingVersion OR version=checkedInVersion)) LIMIT 50
```

Example 3.30. Transformation of a legacy query

On the command line, the query string will usually be surrounded by quotes, so that the shell interprets it as a single argument. You may also have to escape quote characters inside the query string.

The next code block shows the formal definition of the UAPI query language using EBNF:

```
query ::=
    conditional_expression [ order_by ] [ limit ]
    ;
order_by ::=
```

Example 3.31. EBNF definition of the query language

```

    ORDER BY order_entry { "," order_entry }
;
limit ::=
    LIMIT numeric_literal
;
order_entry ::=
    property [ ASCENDING | ASC | DESCENDING | DESC ]
;
conditional_expression ::=
    TYPE ["="] type { "," type } [":" conditional_expression]
    | conditional_expression OR conditional_expression
    | conditional_expression AND conditional_expression
    | NOT conditional_expression
    | "(" conditional_expression ")"
    | BELOW content
    | REFERENCES content
    | property REFERENCES content
    | REFERENCED BY version
    | property IS [NOT] NULL
    | comparison_expression
    | contains_expression
    | value_expression
;
type ::=
    identifier
;
comparison_expression ::=
    value_expression comparison_operator value_expression
;
comparison_operator ::=
    "=" | ">" | ">=" | "<" | "<="
;
contains_expression ::=
    property CONTAINS literal_expression
    | property CONTAINS EXACT literal_expression
    | property CONTAINS PREFIX literal_expression
    | property CONTAINS STEM literal_expression
;
value_expression ::=
    property
    | literal_expression
;
property ::=
    implied_property
    | identifier
;
content ::=
    literal_expression
;
version ::=
    literal_expression
;
literal_expression ::=
    string_literal
    | numeric_literal
    | boolean_literal
    | DATE string_literal
    | PATH string_literal
    | USER string_literal
    | ID string_literal
    | input_parameter
;
boolean_literal ::=
    TRUE
    | FALSE

```

;

The next table shows the definitions of the identifiers and literals of the query language.

Table 3.41. Identifiers and literals

Item	Example	Description
Identifier	Article "parent"	Identifiers consist of Java identifier characters. Where the name of an identifier collides with a keyword or an implied property, the identifier can be enclosed in double quotes to preserve its meaning as an identifier.
String literal	'Title text'	String literals are delimited by single quotes. A single quote inside a string literal is represented by two successive single quotes.
Numeric literal	1234	Numeric literals conform to Java syntax. Essentially, a numeric literal is a sequence of digits, optionally preceded by a minus sign.
DATE literal	'2004-09-08T13:47:07-02:00'	The string used for DATE literals has to be of the form recognized by the DateConverter class (see the Javadoc).
PATH literal	'/MenuSite/Fish'	PATH literals denote a content by giving its path, beginning at the root folder. It is an error if no content exists at the given path.
USER literal	'admin' 'paul@msad'	USER literals denote a user name and a domain name separated by a "@" character. If the domain name is empty, the "@" character may be omitted.
ID literal	'core-media:///cap/content/1'	ID literals denote a content, version or user by giving its ID, as returned by <code>CapObject.getId()</code> .

Queryapprove

The `queryapprove` tool is a combination of the query and the approve tool. You can start a synchronous, structured query against the content repository and approve the resulting resources.

```
usage: cm queryapprove -u <user> [other options]
       [-i] [-e] [-l <limit>] -q <query>

available options:
```

Queryapprove usage


```

-d,--domain <domain name>    domain for login (default=<builtin>)
-e,--versions                 query all versions
-i,--includepath              include unapproved parent folders in
approval
-l,--limit <limit>           maximum number of results to approve
-p,--password <password>     password for login
-q,--query <query>           query string
-qf,--queryfile <queryfile> query file location
-u,--user <user name>         user for login (required)
-url <ior url>                url to connect to
-v,--verbose                  enables verbose output
    
```

Parameter	Description
-e	If set, the query searches for all versions of the content items.
-i	Include unapproved parent folders automatically into approval
-l <limit>	Limit the number of results to the given value.
-q <query>	The query string.
-qf <queryfile>	A file that contains the query string.

Table 3.42. Parameters of the `queryapprove` utility

The following example shows how to approve all content items below the `/MenuSite/Fish` folder:

```
cm queryapprove -u admin -p admin -i -q "BELOW PATH '/MenuSite/Fish'"
```

If folders `/MenuSite` and `/MenuSite/Fish` are not yet approved they will be automatically approved using the `-i` switch.

Querypublish

The `querypublish` tool is a combination of the `query` and the `publish` tool. You can start a synchronous, structured query against the content repository and publish the resulting resources.

```
usage: cm querypublish -u <user> [other options]
       [-e] [-l <limit>] -q <query>
```

available options:

```

-d,--domain <domain name>    domain for login (default=<builtin>)
-e,--versions                 query all versions
-l,--limit <limit>           maximum number of results to publish
-p,--password <password>     password for login
-q,--query <query>           query string
-qf,--queryfile <queryfile> query file location
-u,--user <user name>         user for login (required)
-url <ior url>                url to connect to
    
```

Querypublish usage

```
-v,--verbose          enables verbose output
```

Table 3.43. Parameters of the querypublish utility

Parameter	Description
-e	If set, the query searches for all versions of the content items.
-l <limit>	Limit the number of results to the given value.
-q <query>	The query string.
-qf <queryfile>	A file that contains the query string.

The following example shows how to publish all content items below the /MenuSite/Fish folder:

```
cm querypublish -u admin -p admin -q "BELOW PATH '/MenuSite/Fish'"
```

Search

The search utility invokes the full-text search to find content in the repository.

Example 3.32. Usage of search utility

```
usage: cm search -u <user> [-p <password>] [-d <domain>]
       [-url <ior url>]
       [-q <query>] [-t <contenttype>] [-f <folder>]
available options:
-d,--domain <domain name>      domain for login
                                (default=<builtin>)
-f,--folder <folder>          folder including subfolders
-p,--password <password>      password for login
-q,--query <query>            query string
-t,--contenttype <contenttype> filter by contenttype
                                (including inherited types)
-u,--user <user name>         user for login (required)
-url <ior url>                 url to connect to
-v,--verbose                    enables verbose output
```

Table 3.44. Parameters of the search utility

Parameter	Description
-t	Only search for content items of the given content type and inherited types.
-q	The string to search for.
-f	Only search in the given folder and its subfolders.

3.15 JMX Management

The *CoreMedia Content Server* provides JMX access for management and monitoring. Read the following chapters for further information:

- In the *CoreMedia Operations Basics Manual* read the *Basics of Operations/JMX Management* chapter with general information about JMX and its configuration in CoreMedia applications.
- Read [Section 5.7, “Managed Properties” \[259\]](#) in order to get an overview of the managed properties of the *Content Server*.

Note that configuration changes made via JMX are not persisted, they are effective only until the next server restart. Reloading from file will be disabled for the changed properties.



3.16 User Administration

User management is an essential part in distributed collaborative applications. The two main tasks of the *CoreMedia CMS* user administration are:

→ User Authentication

Defines which user may log on to the *Server*. A generic framework based on the Java Authentication and Authorization Service (JAAS) allows you to authenticate users from arbitrary sources. A concrete solution exists for users and groups from LDAP servers.

→ User Rights Management

Manage users, groups and rights on CoreMedia resources. As with user authentication users and groups may originate from arbitrary sources. Support is provided for LDAP servers, especially *Microsoft Active Directory* servers. The tool to assign rights on resources to groups is the *CoreMedia User Manager*, which is carefully designed to cope with thousands of users and groups sometimes found in large LDAP directories.

3.16.1 Predefined Users and Groups

After a *Content Server* has been started, a set of standard groups with standard users exist. These groups and users are necessary for operation of the *CoreMedia CMS* components, for example the user *workflow* is used by the *Workflow Server* to connect to the *Content Management Server*. When uploading default workflow definitions, workflow groups are created.

Standard Groups and Users

The following table shows the standard groups:

Groups	Description
administrators	Group with administration rights. Only members of this group are allowed to create or delete users and groups.
system	Reserved for system users, do not use for other users
importer	Reserved for importer clients only
approver	Only has APPROVE right
publisher	Only has PUBLISH right
chief editor	Has all rights except SUPERVISE
editor	Only has READ, EDIT and DELETE rights

Table 3.45. Standard groups

These groups are displayed in the following table with their users for different CM server types:

Content and Live Servers:

Groups	Users
administrators	admin
system	webserver
	publisher
	serverdump
	watchdog
	workflow
	feeder

Table 3.46. Users and their groups

Content Management Server:

Groups	Users
importer	importer
approver	no default user
publisher	no default user
chief editor	no default user
editor	no default user

Table 3.47. Users and their groups in Content Management Server only

The default mapping of user rights to users, resources and resource types is as follows:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
ad-minis-trat-ors	/	all	X	X	X	X	X	X	
		+	X			X	X	X	X

Default mapping of user rights
Table 3.48. Default mapping of user rights

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
system	/	all	X	X	X	X	X	-	
		+	X			X	X	-	X
importer	/	all	X	X	X	X	X	-	
		+	X			X	X	-	X
approver	/	all	-	-	-	X	-	-	
		+	-			X	-	-	-
publisher	/	all	-	-	-	-	X	-	
		+	-			-	X	-	-
chief editor	/	all	X	X	X	X	X	-	
		+	X			X	X	-	X
editor	/	all	X	X	X	-	-	-	
		+	X			-	-	-	-
editor	/System	all	X	X	-	-	-	-	
		+	X			-	-	-	-

Workflow Role Groups

The CoreMedia Workflow installation comes with the following predefined workflows which cover the publication of resources:

- ➔ simple-publication
- ➔ two-step-publication

The workflows define workflow roles. Each role is mapped to a group:

Group	Description
composer-role	A user of this group may start a workflow and create a change set
approver-role	A user of this group may approve resources in a change set
publisher-role	A user of this group may publish resources in a change set

Table 3.49. Workflow role groups

A user must be a member of a role group to execute the respective task. The groups are automatically created when a workflow is uploaded and have no user rights. To approve or publish a user must also be a member of another group which has the rights on resources to be approved or published. To approve a content item A in a workflow, for example, the user must be a member of

- approver-role to be able to accept the approve task in the workflow and
- a group which has APPROVE rights on the content item A

In an LDAP-enhanced CoreMedia system, Workflow role groups are mapped in the `workflowserver.properties` file.

```
# remap roles for default workflows
#roles.map.approver-role=approver@example.org
#roles.map.composer-role=composer@exmaple.org
#roles.map.publisher-role=publisher@example.org
```

Example 3.33. Groups in `workflowserver.properties`

These are looked up only once when uploading the workflow. See the [Workflow Manual] for a more detailed description of `workflowserver.properties`.

3.16.2 User Rights Management

CoreMedia CMS provides a fine grained access control which respects group memberships, the folder structure and the resource type hierarchy. Some term definitions are necessary to explain user rights management in more detail:

- **Resource:** A resource is a content item or folder in the CM repository.
- **Resource type:** A resource type defines the fields and the field types of a resource.
- **Folder type '+':** The folder type is a special value needed to define rules on folders.
- **User:** A user may operate on resources, if he has sufficient rights. A user is member of one or more groups.

- **Group:** A group can have users and other groups as members. A group that is member of another group is called a subgroup. A group that has a group as its member is called a super group.
- **Right:** A right is a permission type. Each right allows only some sorts of resource operations. The following table lists the different rights, the assignable resources (folder or content item) and the possible resource operations:

Right	Assignable to	Description
READ	Folder, content item	read content names, content items content and folder names
EDIT	content item	create, check out, check in, rename, move and save content items
DELETE	content item	mark and unmark a content item for deletion, move an item to trash
APPROVE	Folder, content item	approve, disapprove, approve place, disapprove place a content item or folder
PUBLISH	Folder, content item	publish a content item or folder
FOLDER	Folder	create subfolder, rename, move and delete a folder
SUPERVISE	content item	check in or uncheckout a content item from a different user, grant new rights

Table 3.50. User rights

- **Rule:** A rule defines a right on a resource of a certain resource type. A rule is granted not to a user but to a group. A user must be a member of a group to get the rights of the group. So a rule consists of a group, a resource, a resource type and a right parameter. Formally a rule is a four-tuple

$r = (gr, rs, rt, rg)$ from $(GROUPS \times RESOURCES \times RESOURCE TYPES \times RIGHTS)$

where

GROUPS is the set of groups

RESOURCES is the set of resources

RESOURCETYPES is the set of resource types and the special folder type "+"

RIGHTS is one of (READ, EDIT, DELETE, APPROVE, PUBLISH, FOLDER, SUPERVISE)

Rule Tables

The following sections will often show rules to explain user rights. The rules are displayed in a rule table. Here is an example of a rule table:

Example rule table
Table 3.51. Example rule table

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

A rule table is closely related to the rule table in the *CM User Manager*. The table is composed of the following columns:

- **Group:** The group, a right is assigned to. If no value is set, the value in the next column cell above is assumed.
- **Resource:** The content item or folder a right is assigned to. The table cell contains the path to the content item (the name written in lowercase letters) or folder (the name starting with an uppercase letter). If no value is set, the value in the next column cell above is assumed.
- **Resource Type:** The resource type or the folder type +, a right is assigned to. If no value is set, the value in the next column cell above is assumed.
- **Right columns:** reserved for rights with the following meaning:

X: the right is set

-: the right is not set

no entry: the right cannot be set. FOLDER rights cannot be set if the Resource Type column contains a resource type. It can be set if the Resource Type column contains the folder type +. EDIT and DELETE rights cannot be set if the Resource Type column contains the folder type +. The two rights can be set for ordinary resource types.

The rule table example above defines two rules for a group G and a folder F1:

1. Group G has READ and EDIT right on the folder F1 for all content items of content type Article.
2. Group G has READ right on the folder F1 for the folder type.

The following section specifies in more detail what resource operations follow from a right on a content item and folder.

Required Rights for Resource Operations

A right is a permission type and each right allows only some sorts of resource operations. Some operations require several rights, moving a content item, for example. The following table lists required rights for operations on a content item *doc* of type *Article* in Folder *F2*:

Operations:

- read fields of content item *doc*
- read implied properties of content item *doc* like date of last modification

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	X	-	-	-	-	-	

Table 3.52. Rule to read a content item

Operations:

- create new content item *doc* in folder *F2*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-	X	-	-	-	-	

Table 3.53. Rule to create a content item

Operations:

- rename content item *doc*
- save content item *doc*
- checkout content item *doc*
- check in or uncheckout content item *doc* if the same user has checked it out before.

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	X	-	-	-	-	

Table 3.54. Rule for content item operations

Operations:

- move content item *doc* from folder *F2* to Folder *F3*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-	X	-	-	-	-	
G	/F1/F3	Article	-	X	-	-	-	-	

Table 3.55. Rules to move a content item

Operations:

- mark content item *doc* for deletion
- unmark content item *doc* for deletion

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	-	X	-	-	-	

Table 3.56. Rule to mark or (un)mark a content item for deletion

Operations:

- move content item *doc* from folder *F2* to trash

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	-	X	-	-	-	
	/F1/F2	Article		X					

Table 3.57. Rules to delete a content item

Operations:

- approve content item *doc*
- disapprove content item *doc*
- approve place content item *doc*
- disapprove place content item *doc*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	-	-	X	-	-	

Table 3.58. Rule to (dis)approve a content item

Operations:

- publish content item *doc*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	-	-	-	X	-	

Table 3.59. Rule to publish a content item

Operations:

- check in or uncheckout content item *doc* for a user different from the one who checked out the content item

Required right:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	-	-	-	-	X	

Table 3.60. Rule to check in content items of other users

The following paragraphs list required rights for certain operations on a folder *F2* in parent folder *F1*:

Operations:

- read implied properties of folder *F2*, like date of last modification or names of children

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	+	X			-	-	-	-

Table 3.61. Rule to read folder properties

Operations:

- place approve folder *F2*
- place disapprove folder *F2*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	+	-			X	-	-	-

Table 3.62. Rule to place approve or disapprove a folder

Operations:

- publish folder F2

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	+	-			-	X	-	-

Table 3.63. Rule to publish a folder

Operations:

- create a subfolder in folder F2

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	+	-			-	-	-	X

Table 3.64. Rule to create subfolders

Operations:

- rename folder F2
- mark folder F2 for deletion
- unmark folder F2 for deletion

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	+	-			-	-	-	X

Table 3.65. Rule to operate on subfolders

Operations:

→ move folder F2 to folder F3

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	+	-			-	-	-	X
	/F3	+	-			-	-	-	X

Table 3.66. Rules to move a folder

The root folder has special rights. You cannot move, delete or rename the root folder.



Above you saw that the SUPERVISE right is necessary for non-administrator users to check-in content items of other users. Now you will see that the SUPERVISE right is the right for a non-administrator group to grant new rights:

Operations:

→ users of group G grant rights on content item doc for resource type Article

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-			-	-	X	-

Table 3.67. Rule to supervise a content item

Operations:

→ users of group G grant rights on folder F2 for resource type Article

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-			-	-	X	-

Table 3.68. Rule to supervise content items in a folder

Operations:

→ users of group G grant rights on folder F2 for the folder type +

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	+	-			-	-	X	-

Table 3.69. Rule to supervise a folder

You do not have to define rules for each group, resource or resource type. A rule definition may contain a

- **super group:** the rule is applicable for all subgroups
- **super folder:** the rule is applicable for all subfolders
- **super type:** the rule is applicable for all subtypes

When using super groups, super folders and super types the number of rules is greatly reduced but the problem of conflicting rules emerges. The problem appears when two rules for a super group and subgroup or a super folder and a subfolder or a supertype and a subtype are defined. The following section explains how rights for a resource are evaluated from a set of rules and how conflicting rules are resolved.

Computation of Rights

This section will explain in detail how the rights for a resource are computed from a set of rules. First it is defined if a rule is applicable. A rule is *applicable* if it is involved in the computation of rights for a certain operation. Look at the following rule table described earlier:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

Table 3.70. Example rules for rights computation

In the following cases the first rule in the rule table is not applicable:

1. A user of a different group *G2* which is not a subgroup of *G* wants to operate on a resource.
2. A user of group *G* (or a subgroup) wants to operate on a content item in a different folder *F2* which is not a subfolder of *F1*.
3. A user of group *G* (or a subgroup) wants to operate on a content item *teaser1* in folder *F1* (or a subfolder). Content item *teaser1* has the content type *Teaser*, which is not a subtype of *Article*.

The first rule is applicable only, if

1. the user is member of group *G* or a subgroup and
2. the user operates on a resource in folder *F1* or a subfolder of *F1* and
3. the content item has type *Article* or a subtype of *Article*.

It is now possible that two or more rules are applicable to a resource. Have a look at the next rule table:

Table 3.71. Example for conflicting rules

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G1	/F1	Article	X	X	-	-	-	-	
		+	X			-	-	-	-
G2	/F1	Article	X	-	X	-	-	-	
G1	/F1/F2	Article	X	-	-	X	-	-	
		+	X			-	-	-	-
G1	/F1	ShortArticle	X	X	-	-	X	-	

Let's assume the following:

- G2 is a subgroup of G1,
- F2 is a subfolder of F1 and
- the content type *ShortArticle* is a subtype of *Article*.

Users of group G2 have no EDIT rights on articles in folder F1 but DELETE rights. In subfolder F2 there are no EDIT and DELETE rights for articles but APPROVE rights. And finally there are no DELETE and APPROVE rights for content items of type *ShortArticle* in F1, but READ, EDIT and PUBLISH rights. There are lots of conflicting situations, for example:

1. A user of group G2 wants to edit or delete an article content item directly in folder F1.
2. A user of group G1, not G2, wants to edit, approve or delete an article content item in subfolder F2.
3. A user of group G1, not G2, wants to edit, approve, delete or publish a short article content item in subfolder F1.

These conflicts are resolved by the definition, that a more specific rule takes precedence over a less specific rule. A rule *r1* is *more specific than* a rule *r2* if and only if

SP1.) The group in rule *r1* is a subgroup of the group in *r2*

SP2.) The groups are equal and the resource in rule *r1* is located inside the folder of rule *r2*

SP3.) The groups and the resources are equal and the resource type in rule *r1* is a subtype of the resource type in rule *r2*

Rules are not merged as can be seen from the definition. If you apply the definition, you get the following conflict resolutions for the three examples above:

1. *G2* is a subgroup of *G1*. From *SP1* it follows that the user in group *G2* who wants to edit or delete an article content item directly in folder *F1*, has the rights to READ and DELETE, but not to EDIT.
2. *F2* is located in *F1*. From *SP2* it follows that the user in group *G1* who wants to edit, approve or delete an article content item in subfolder *F2*, has the rights to READ and APPROVE, but not to EDIT and DELETE.
3. *ShortArticle* is a subtype of *Article*. From *SP3* it follows that the user, who wants to edit, approve, delete or publish a short article content item in subfolder *F1*, has the rights to READ, EDIT and PUBLISH, but not to DELETE and APPROVE.

A rule that is preceded by another rule is said to be *shaded*. A rule is called *effective* if it is applicable and not shaded. The *effective rights* of a group, a resource and a resource type are the union of the rights of the effective rules. To explain this, look at the following rule table:

Table 3.72. Example rules to compute effective rights

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
		+	X			-	-	-	-
G	/F2	Article	X	-	-	X	-	-	
		+	X			-	-	-	-

If a user is member of group *G* then the effective rights for the two folders *F1* and *F2* are unified, so the user can read articles in both folders, edit articles in *F1* and approve articles in *F2*. Of course the user cannot edit articles in *F2* nor can he approve articles in *F1*.

The effective rights are nearly the rights of a group on a resource for a resource type. There are only three exceptions:

1. **Navigate Through:** If there are no effective rules for a Folder *F1* and the group has non-empty effective rights for a resource located beneath *F1* then the group has implicit READ rights for *F1* and the folder type "+". This sounds more complicated than it is. Look at the simple example:

Table 3.73. Example rules with implicit navigate through right

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

The user in group *G* can edit the article in folder *F2*. There is an implicit navigate through right for folder *F1*. The example above is equivalent to:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	+	X			-	-	-	-
G	/F1/F2	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

Table 3.74. Example rules with resolved navigate through right

2. If the effective rights are not empty, the group also has the implicit READ right for any resource and resource type:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	-	X	-	-	-	-	
		+	X			-	-	-	-

Table 3.75. Example rules with implicit READ right

In the example above the user in group *G* has implicit READ right for an article in *F1*. This is equivalent to:

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

Table 3.76. Example rules with explicit READ right

- If a group has no READ rights on a parent folder for folder type "+", then a child folder has no READ rights at all. The READ right can only be withdrawn explicitly by a rule with empty rights.

Table 3.77. Example rules with READ right withdrawn

Group	Re-source	Re-source Type	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	+	-			-	-	-	-
G	/F1/F2	Article	X	X	-	-	-	-	
		+	X			-	-	-	-

A user in group G does not have READ rights on folder F2 because there is no READ right for the parent folder F1. To grant READ rights, the right must be explicitly set in the first row or the first row must be removed.

The effective rights of a group on a resource for a resource type with the three exceptions above are displayed in the resource info window in the *Site Manager* in the rights tab.

3.16.3 Administrator Groups

Members of administrator groups are supposed to do administrative work in the CoreMedia system. Therefore, these members have special privileges:

- See all workflows
- See all rules attached to a resource and differentiate between Live Server groups and Content Server groups.
- Check-in and check-out resources of all users
- Direct approval and publication
- Create new users and groups
- Create wide links in a multi-site environment

Right after installation there exists only one administrator group with one user; the group *administrator* with the user *admin*. It is not possible to revoke the administrator flag from this group in order to prevent the admin user from lock out. From all other administrator groups you are able to revoke this flag.

3.16.4 Content Server Groups and Users

CoreMedia CMS has some groups and users for standard operation (so called built-in users/groups see [Section 3.16.1, “Predefined Users and Groups” \[160\]](#), such as workflow) on the *Content Management Server*. You can add additional users and groups to administrate access rights on the content. *CoreMedia* distinguishes between content and live groups. Groups which have rules valid on the *Content Management Server* and groups which have rules valid on the *Live Server* respectively.

Example:

You create a news site which offers content in the categories sports, politics, economy and gossip. Your editorial staff contains 20 editors, 5 for each category. Each editor has only access to content of his specific field. So you need to administrate at least 4 additional groups and 20 users on the *Content Management Server* each group with different access rights.

You can either administrate these users and groups using the built-in user administration of the *Site Manager* or you can connect the *CoreMedia* system to an existing LDAP server. Therefore, *CoreMedia CMS* supports any LDAP server. Because LDAP has no obvious concept for content and live groups *CoreMedia CMS* provides a `UserProvider` class (see [Section 3.13.3, “LdapUserProvider” \[85\]](#) and the Javadoc). This class differentiates between live and content groups. *CoreMedia* provides the predefined `ActiveDirectoryUserProvider` to connect to an Active Directory server. If you use an Active Directory server you have the possibility to define all groups of this server as *Live Server* groups, *Content Management Server* groups or both using the properties

```
com.coremedia.ldap.contentgroups=true
com.coremedia.ldap.livegroups=true
```

in the `WEB-INF/properties/corem/jndi.properties` file.

If you want to connect to another LDAP server you can extend the `LdapUserProvider` class for your own user provider (see [Section 3.13.3, “LdapUserProvider” \[85\]](#) and the Javadoc).

Groups need to be connected with rules in order to have impact. In the example above, the group sport might have a rule which allows a member to read and write content from and into the sports folder. Use the *User Administration Window* of the *Site Manager* to add rules to your groups. Read [Section 3.16.2, “User Rights Management” \[163\]](#) for details on rights and rules.

3.16.5 Live Server Groups and Users

CoreMedia CMS needs some groups and users for standard operation (so called built-in users/groups see [Section 3.16.1, “Predefined Users and Groups” \[160\]](#), for

example publisher) on the *Live Server*. These groups and users are created when the server starts for the first time. In addition to these built-in groups you can add business oriented users and groups. With these groups you administrate access rights on the delivered content. You define the users who are allowed to read content.

Example:

You create a news site which offers standard content for registered customers, gold content for paying customers and platinum content for customers paying even more. The customers might sum up to 100.000. So you need to have at least 3 groups (such as standard, gold, platinum) on the *Live Server* containing 100.000 user.

It's as likely as not that your company administrates these users on an LDAP server. Therefore, *CoreMedia CMS* supports any LDAP server. Because LDAP has no obvious concept for content and live groups *CoreMedia CMS* provides a `UserProvider` class (see [Section 3.13.3, "LdapUserProvider" \[85\]](#) and the Javadoc). This class differentiates between live and content groups. *CoreMedia* provides the predefined `ActiveDirectoryUserProvider` to connect to an Active Directory server. If you use an Active Directory server you have the possibility to define all groups of this server as *Live Server* groups, *Content Management Server* groups or both using the properties

```
com.coremedia.ldap.contentgroups=true  
com.coremedia.ldap.livegroups=true
```

in the `WEB-INF/properties/corem/jndi.properties` file.

If you want to connect to another LDAP server you can extend the `LdapUserProvider` class for your own user provider (see [Section 3.13.3, "LdapUserProvider" \[85\]](#) and the Javadoc).

You must not change the `LdapUserProvider` after starting the content server with LDAP user authentication. If, for example, you have defined content groups first, and change this to content and live groups later, the live server will not notice this change.

Groups and users or memberships of groups and users are not published or replicated so you have to administrate them individually on each server. There is only one exception to this rule:

Assume, you have a resource with a rule connected to a group and this group does not exist on the *Live Server*. If you publish this resource, a group with the same name will be created on the *Live Server*. This group has no members and is no member of another group. It's only a placeholder so that the rule is connected to something, which you have to populate with memberships.

Groups need to be connected with rules in order to have impact. In the example above, the group platinum might have a rule which allows members to read content contained in the platinum folder. For your convenience rules are administrated on the *Content Management Server* and are published and replicated. Therefore, the *Content Management Server* needs to know the groups used on the *Live Server*. The easiest way to achieve consistency between the two server types is to use the same LDAP server with the same `UserProvider` configuration. You can also use different LDAP server but you have to ensure that the Live Groups on *Live Server* and *Content Management Server* are the same. Groups are identified by their name and domain so this has to be identical on both servers.

If you have provided the *Live Server* groups to the *Content Management Server*, you can use the *User Administration Window* of the *Site Manager* to add rules to the groups. *Live Server* groups are identified by their checked Live Server Group checkbox. A rule will appear on the *Live Server* not until a resource connected to the rule has been published. To maintain data integrity only READ rights are allowed on the *Live Server*.

3.16.6 Managing Users

The User Manager window is the default window for the user admin, all other administrator group members can access the functionality by choosing **Window | User manager** window from the *CoreMedia Editor's* explorer window.

The User Manager interface

The new interface holds users and groups on tabs.

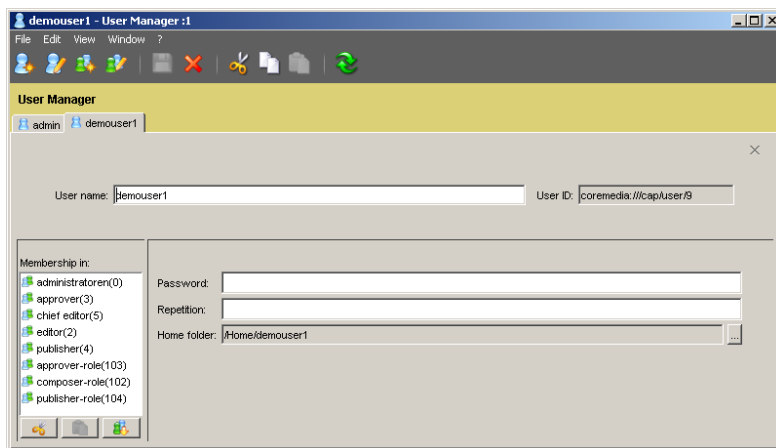


Figure 3.3. User administration window

Creating New Groups

Only members of *Administrator* groups are entitled to create new users and groups. For Administrator groups, the *Administrator group* flag is set as shown in the next figure.

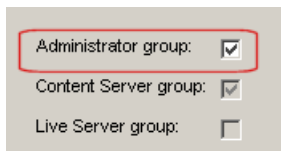



Figure 3.4. Administrator Group

A new group is created by giving it a new name and allocating existing users to the group. The *Group id* is automatically given by the application.

Create a group

1. Choose **File|New group**


- or click  on the tool bar
- or press <CTRL>+<G> resp. <ALT>+<G>

2. Name the group.

3. Choose the group type.

You may choose *Administrator group* to provide administrator rights to the group members or *Content Management Server group* to create a non administrative group on the *Content Management Server*. Check *Live Server group* if you want to create a group for the *Live Server* (you have to create the same group on the *Live Server*, read [Section 3.16.5, “Live Server Groups and Users” \[178\]](#) for details on Live Server user/groups).

4. Save the group settings.

- click  on the tool bar.
- or press <CTRL>+<S> resp. <ALT>+<S>

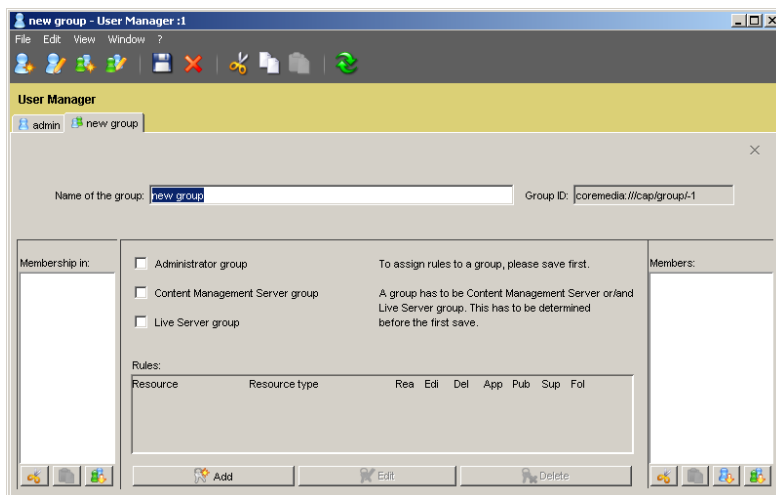


Figure 3.5. Creating a new group

After creating a group, you may continue with either adding members to groups, or assigning rules to groups.

In order to add members to a group:

1. Open the group in a tab
2. Choose **Edit | Members** and users or groups that you want to add as members to your group.

Now you may continue with assigning rules to resources:

3. Open the *group* tab
4. Click on the **[Add]** button at the bottom of the window.

To assign rules to a group

1. Choose a resource after that a resource type (depending on your SUPERVISE rights on this resource).

If you don't have SUPERVISE rights on that resource, you cannot choose a resource type. You can assign rights to a folder by choosing "Folder" in the field *Resource Type*.

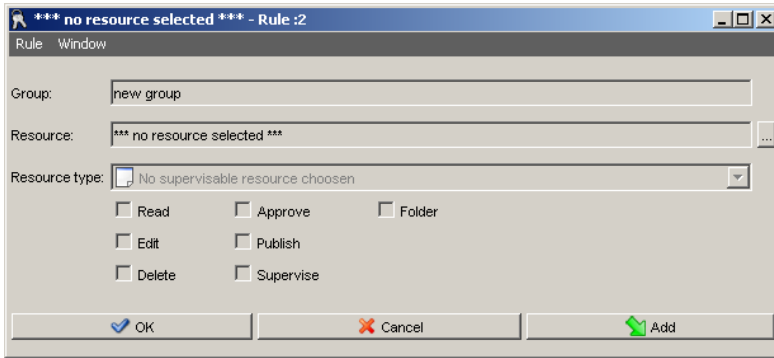



Figure 3.6. Assign rules for group

The properties of a group can be characterized by the combination of multiple resources with their special rights.

Creating New Users

To create a new user, a new user profile is created by means of the  symbol.

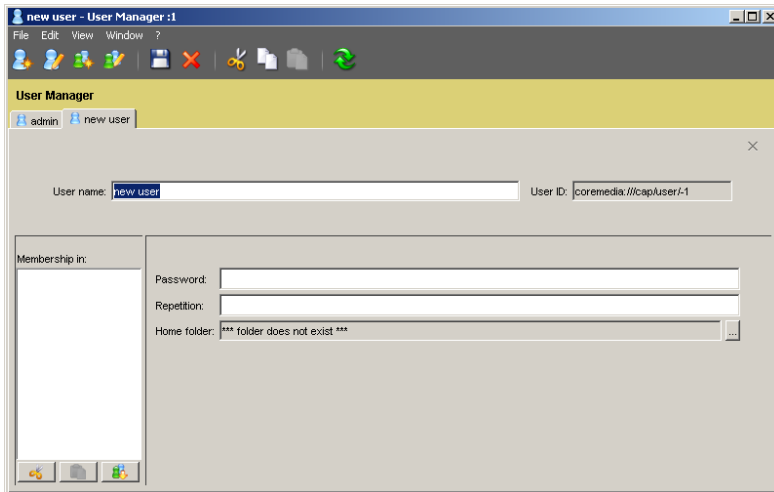


Figure 3.7. Creating a user profile

A name and a password must be assigned to the new user.

A home folder stores personal content items, as well as queries, the user-defined dictionary, *Site Manager* settings or draft content items of the user. If you do not enter a folder, it will be created automatically in the /Home folder, with the name of the user as the folder name.



By default, a user has no rights on its home folder. You have to add the user to a group which has appropriate rights on this folder. See for example the pre-defined editor group.

If a folder with this name already exists, a new name will be created according to the following scheme:

1. A counter which counts from 1 to 1000 will be used to attach a unique number to the user name `<UserName>_(<No.>)`.
2. If 1. fails, the time elapsed since 1.1.1970 in ms will be attached to the user name: `<UserName>_(<TimeInMs>)`.
3. If 1. and 2. fail, an error message appears and no folder will be created. This might indicate problems with rights or the network.

You may create your own naming pattern by adding an entry for the property `save-user-homeFolderNamePattern` to your custom editor bundle. Read the localization chapter of the *Editor Developer Manual* for more details on own bundles. You can use these variables to create unique names:

- {0}: the user name
- {1}: a counter which counts from 1 up to 1000 until a unique name is found or the threshold is reached
- {2}: the time elapsed since 1.1.1970 in ms

Example:

The editor bundle `myBundle.properties` contains the following entry:

```
save-user-homeFolderNamePattern=UserFolder_{1}
```

The User Manager will create home folders `UserFolder_1`, `UserFolder_2` ... `UserFolder_1000`. If The User Manager is not able to create new folders according to this scheme, the escalation scheme 1. - 3. described above will be used

Alternatively, you can create a home folder by your own, choosing any desired name. Nevertheless, you should create the folder in the `/Home` folder.

You can allocate one or more groups to the user determining his rights in the CoreMedia system.

Every user can change his own password later. Furthermore, all users can view the rights structure of the different groups in User Administration but can not change it.

Since the user name is involved in password encryption, it is not possible to change a user name. To do so, you have to create a new user and delete the old one.

Viewing Resource Rights

The rights of the groups of the current user for a certain resource can be seen in the **Rights** tab in the properties of the resource. Here you will see the merger of all rights provided by all groups the user belongs to, specified by the resource types.

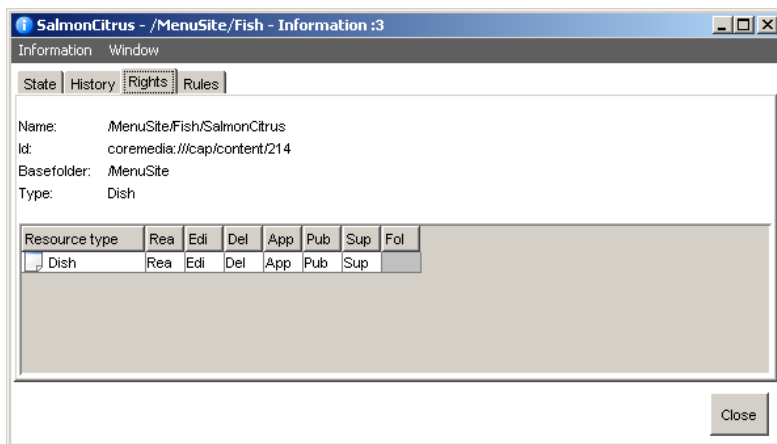


Figure 3.8. Rights structure of a resource

Using the tab **Rules**, the applicable rules of all groups of the logged in user for the current resource are shown.

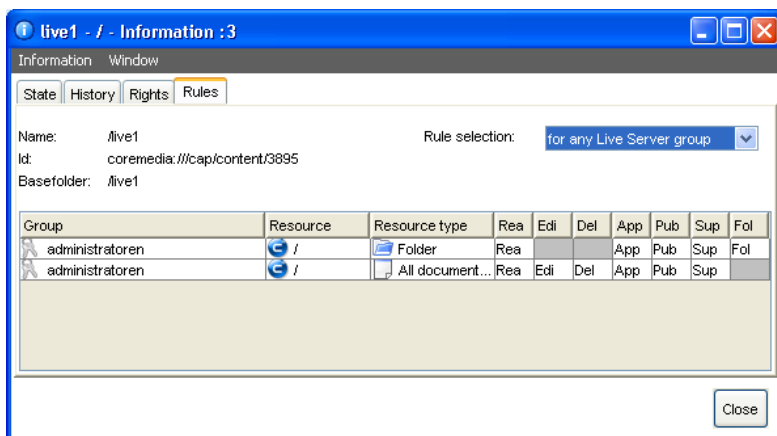


Figure 3.9. Rules of the user for a resource

The administrator can use the *Rule selection* dropdown box to change the view:

- *for current user*: Shows all rules attached to the groups of the current user which are valid for this resource.
- *for any Content Server group*: Shows all rules attached to Content Server groups which are valid for this resource.
- *for any Live Server group*: Shows all rules attached to Live Server groups which are valid for this resource.
- *defined on this resource*: Shows all rules directly defined for this resource.

If a rule is applied to content types in a directory, the current rights structure for the particular group is passed on to the content items in subdirectories. If another rule should apply in a subdirectory it has to be created explicitly for the particular content types. For available rights and their impact see [Section 3.16.2, “User Rights Management” \[163\]](#).

If a user possesses different rights for a resource due to belonging to multiple groups, these rights are additively combined.

Example:

Table 3.78. Example groups

User	Group	Directory	READ	EDIT	DELETE	AP-PROVE	PUB-LISH	GRANT	FOLDER
UserA	Sport	/News	x						
UserA	Politics	/News	x	x	x				

User A has READ, EDIT and DELETE rights for the CoreMedia directory `/News`, since the rights of the groups add.

Applying New Rules via File Menu

For administrators, there is an entry in the **File** menu of the explorer window: **New rule**.

This menu item allows you to adjust rights on a selected resource directly.

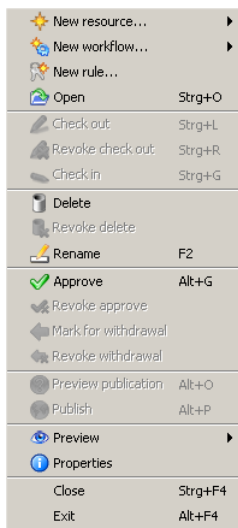


Figure 3.10. The File menu (administrator view)

Assigning Licenses to Users

Because the license limits the number of concurrently logged in users, it may happen that this limit is reached and additional logins must be rejected. This is done on a first come, first served basis: whoever logs in first is granted permission, later attempts may fail. When you use *CoreMedia CMS* to manage the content of a single enterprise, this is normally the desired behavior. When it is important that a new connection is opened, there is always the possibility to ask other users to log out, thereby freeing concurrent licenses.

When hosting multiple content applications in a single Content Management Environment, you might want to control the logins more closely. You may have legal or operational requirements that a login is possible for certain users. Because it is not generally possible to free already assigned licenses, the server must take care to reserve licenses when they are still available.

For example, you might want to ensure that each of your five content applications is granted at least one login. Other licenses should be freely distributable. If you have bought eight concurrent licenses, there must not be two content applications that lock up three sessions each, because there are only two concurrent sessions for three content applications left. Still, there is nothing wrong with *one* content application that reserves three or even four concurrent sessions.

To facilitate the controlled distribution of concurrent licenses, a so-called *login bouncer* can be enabled. Picture it as guarding the entry to the server based on owner-defined rules. In order to enable a bouncer, set the property `cap.server.login.bouncers` of the file `contentserver.properties` to the location

of an XML file that describes the bouncer's configuration. Typically, such files are placed in the directory `properties/corem` below the installation directory.

The bouncer grants licenses from a number of *pools* and enforces certain *limits* to the number of logged in users. Both pools and limits apply to that set of users whose names match a given regular expression and who directly or indirectly belong to certain groups.

A limit sets the maximum number of licenses that is granted to those users that are subject to the limit. A user may be subject to multiple limits. If even one limit is reached, the user is not allowed to log in.

A pool is controlled using two parameters: a minimum size and a maximum size. Given that the limits are respected, the bouncer will try to distribute the users and the available concurrent logins among pools in such a way that:

- every pool receives at least its minimum number of licenses,
- every pool receives at most the number of users given by its maximum size, and
- every pool receives at least as many licenses as users.

Upon every login attempt it is checked whether a new license can be allocated to the pools, possibly reassigning licenses across pools for users that are already logged in.

Minimum sizes guarantee that a certain number of users whose name matches the pools pattern can login. No matter which other users try to log in, these licenses are set aside until a user of that pool demands a license. Maximum sizes allow the number of licenses provided to grow as needed. Contrary to limits, additional users that match a pool's pattern may login after the pool has been exhausted. However, in this case there must be some other applicable pool with free licenses. One notable exception is the user `admin(0)`, who is always permitted to login regardless of the set of currently logged in users. The super admin is only controlled by the license itself, not by the bouncer.

In the XML configuration, you may install bouncers for each individual service by inserting a `<LoginBouncer>` elements below the `<LoginBouncers>` document element. Normally, only the service `editor` is restricted, because other services are used by scripts and other servers in a more deterministic way.

For every bouncer, you can add an arbitrary number of `<Limit>` and `<Pool>` elements. Three attributes define the set of users to which a limit or pool applies. At least one of the three attributes should be given. When more than one is given, the set of users is restricted further.

- `pattern`

A regular expression according to the syntax of the Java package `java.util.regex`. Only users whose names match the pattern are included in the set. The name of a user defined in the built-in user management ends in a `@`. For example, there might be two different users `joe@` and `joe@do` `main`. The pattern must take the domain part into account.

→ `directgroup`

The name of a group whose direct members are included in the set.

→ `group`

The name of a group whose direct and indirect members are included in the set. This restriction can be costly to check in the case of a deep group hierarchy.

For a limit the following additional attribute applies:

→ `max`

The maximum number of concurrent logins allowed by this limit. This attribute is required.

For a pool the following additional attributes apply:

→ `max`

The maximum number of concurrent logins granted by this pool. Defaults to infinity.

→ `min`

The minimum number of concurrent logins granted by this pool. Defaults to 0. The sum of all minimum values must not exceed the number of available concurrent logins.

A sample configuration might contain XML elements as shown in [Example 3.34, "A sample login bouncer configuration" \[189\]](#):

```
<LoginBouncers>
  <LoginBouncer service="editor">
    <Limit pattern=".*@(domain1|domain2)" max="80" />
    <Pool pattern=".*@domain1" min="5" max="50" />
    <Pool pattern=".*@domain2" min="5" max="50" />
    <Pool pattern=".*@domain3" max="50" />
    <Pool pattern=".*@domain4" max="50" />
    <Pool directgroup="administrator@domain1" min="1" />
    <Pool directgroup="administrator@domain2" min="1" />
    <Pool directgroup="administrator@domain3" min="1" />
    <Pool group="administrator@domain4" min="1" />
  </LoginBouncer>
</LoginBouncers>
```

Example 3.34. A sample login bouncer configuration

Assume that a total of 100 licenses is available. The limit line states that at most 80 of these 100 licenses may be acquired by users from either `domain1` or `do`

`main2`. The pools for `domain1` and `domain2` allocate between 5 and 50 licenses. Unless 5 users from one of these domains are already logged in, there are still free licenses to allow another login. The pools for `domain3` and `domain4` do not specify minimum sizes. It is therefore possible that, for example, no user from `domain4` is able to log in to the system, because users from the three other domains are locking up all available licenses.

Finally, the last lines indicate that some licenses should be put aside for the members of the administrator groups of any domain, to be used in emergency cases. For `domain4` you add indirect members of the administrator group to the pool.

Normally, a careful assignment of licenses to pools should suffice for most setups, with limits being used in exceptional cases. The number of pools and limits should be kept relatively low, preferably not exceeding 100 in total, lest the processing time gets significant. Note that groups cannot be indicated by regular expressions as this would lead to a significant reduction in performance.

3.17 WebDAV Support

The WebDAV server supports access to content of the *CoreMedia Content Management Server* with WebDAV-enabled applications like Microsoft Word 2010.

The WebDAV support is provided by a web application which is deployed in the servlet container. The contents of the *Content Management Server* appears in a web folder of the WebDAV client. [Section 3.17.2, “Configuration and Operation” \[194\]](#) describes how to create a Windows web folder. Configure WebDAV support in the files

- `webdav.properties`,
- `capclient.properties`

located in the directory `<WebDAVServletContext>/WEB-INF/properties/coremedia` where `<WebDAVServletContext>` is the path to the WebDAV servlet context, for example `tomcat/webapps/webdav`. The WebDAV web application uses the *CoreMedia logging-component* for logging which already provides default logging configuration. To change the configuration, you can put a `logback.xml` file into the `WEB-INF` directory. For more information about the logging component, see the [CoreMedia DXP 8 Manual].

3.17.1 Concepts

The CoreMedia WebDAV support exports the content of the CoreMedia repository into the file system of your computer; or, to be more exact, the part of the CoreMedia repository on which you have at least read rights. For each folder of the CoreMedia repository you will find a folder in the file system with the same name. The export of CoreMedia content items is a bit more complicated. In general, you define in `webdav.properties` which properties of which content types you want to export. You can do so in two different ways depending on the number of properties you want to export:

- export as files
- export as folders

You can configure the mapping of content items to files or folders in the `webdav.properties` file. See [Section 5.6, “Reference of webdav.properties” \[251\]](#) for a description of all properties.

Export as files

If you only want to export one property of a specific content type, you are better off exporting the items as files. You will get one file in the file system for each

corresponding content items, where the file has the same name as the content item. The exported item is located in a directory structure that is equal to the CoreMedia folder structure. By default, no file extensions are appended to the exported file. You can configure the WebDAV Server to append a file type extension to all content items that are exported as file. The extension is automatically derived from the MIME type of the exported field and must be configured in the file `mime.properties`. If no extension can be derived, the extension `.bin` is used.

If you have more than one field in your content type, which you want to edit via WebDAV, you have to export the content item as folders.

Export as folders

If you want to export more than one property of a content item, you can export it as folders. In this case, you will get for each exported content a folder with the name of the content item. This folder contains one file for every exported property, where the file has the localized name of the property with a file extension that is automatically derived from the MIME type of the exported field and must be configured in the file `mime.properties`.

This export is more flexible than the export as files but it might be more confusing for users because you have WebDAV folders for CoreMedia folders as well as for content items.

Creating content items

When you create a new file with your WebDAV client, the WebDAV server has to determine the content type of the content it should create in the CoreMedia repository. This is done via a mapping between a part of the name of the file or folder (such as the extension) and the content type to create (see [Section 5.6, "Reference of webdav.properties" \[251\]](#)). So, if you have registered `.doc` as an extension for CoreMedia items of type `CMWord` and you create a new Word document "New-Text.doc" in a WebDAV folder, then the WebDAV server will create a content item of type `CMWord` named "NewText" in the CoreMedia repository. You can use WebDAV to rename the content item after its creation, which of course does not change the type of an already created item. The configured extension mapping is only used for the creation of content items.

When you use the folder mapping for content items and want to create a new content that uses folder mapping, you have to create a new Folder with a name that ends with the registered text. The task is a bit more complicated because Windows creates a "New Folder" first before you can specify the name of the folder. This leads to the creation of a CoreMedia folder with the same name. If you immediately rename the still empty folder to a name with an extension that is mapped to a content type, then the empty CoreMedia folder is deleted and a document of the wanted type is created instead. The WebDAV server will automatically create empty files for all exported properties of the new document.

You can also configure the catch-all extension * for one content type to create content items of that type for all names which do not match another configured extension. Note, that if you configure such an extension for a content type which uses the folder mapping, you will not be able to create CoreMedia folders with WebDAV.

The Info File

The root WebDAV folder contains an HTML information file (info.html) that displays some additional information. The information file contains the following:

- The name of the logged in user.
- The exported content types.
- The file extensions associated with the exported content types.
- The locked content items of the user. Members of the administrator group also see content items locked by other users.
- The name of the WebDAV application which locked the content item.
- A list of error messages that occurred during the last WebDAV requests.

Errors occur if you do not have the appropriate rights (HTTP error code 403) or when problems appear during writing, for example when a string is longer than allowed by the String field.

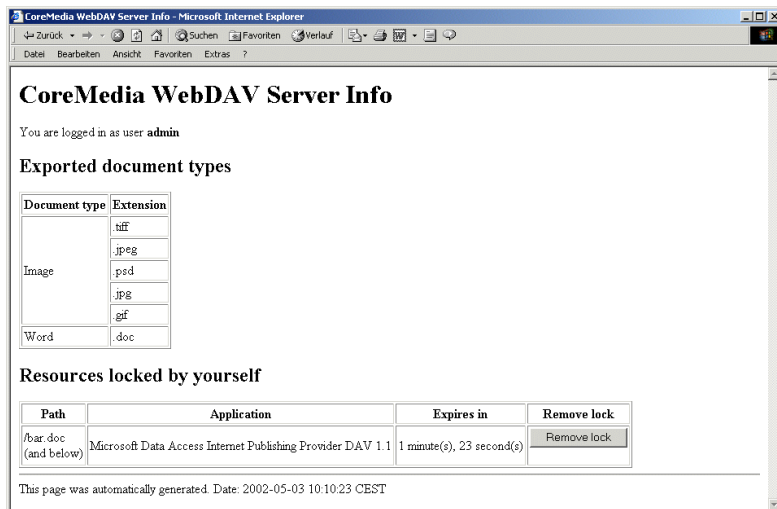


Figure 3.11. Example: WebDAV file

To remove a content item lock click on the appropriate **[Remove lock]** button. Members of the administrator group can also unlock the content items of other

users. This is useful if an application has not unlocked an opened content item, for example because the application has crashed. Normally, it is not necessary to unlock a content item manually.

Double-clicking the `info.html` file on a Windows 7 system, opens it in the Internet Explorer with a special URL. Because of this URL, the buttons to remove locks and messages do not work. As a workaround, you can enter the correct URL (`http://<host>:<port>/info.html` for example) in the IE.



3.17.2 Configuration and Operation

This chapter describes tasks necessary for the configuration and administration of WebDAV. This chapter does not cover all possible tasks. If you miss some tasks, have a look at [Section 5.6, “Reference of webdav.properties” \[251\]](#).

Creating a WebDAV Folder on the Client

The recommended way to use WebDAV with Windows 7 and Windows Vista is to use Basic authentication with an HTTPS connection. Configure this in the `webdav.properties` file:

```
authentication.scheme.1=Basic
```

If you want to use HTTPS, you have to prepare first the Tomcat of the WebDAV server as described in the *CoreMedia Operations Basics Manual*.

In order to create a WebDAV folder on your computer proceed as follows:

Windows 7

1. Open a Windows Explorer window.
2. Click **Map Network Drive** to connect to the network computer. Alternatively choose **Tools|Map Network Drive** from the window menu.

A wizard opens up.

3. Enter the URL of the WebDAV server directly into the *Folder* field.
4. Click **[Finish]**.

Connect using different credentials

Windows tries to connect with the server. If the connection could be established, you are asked for user name and password.

5. Enter the user name and password.

Windows Vista

1. Open a Windows Explorer window.
2. Click **Map Network Drive** to connect to the network computer. Alternatively choose **Tools|Map Network Drive** from the window menu.

A wizard opens up.

3. Enter the URL of the WebDAV server directly into the *Folder* field.
4. Click the *Connect using different user name.* link. A new window opens up.
5. Enter the user name and password and click **[OK]**.
6. Click **[Finish]** in the wizard.

Windows now connects with the server.

You can also use the command line to connect to the WebDAV server:

```
net use <driveLetter>: https://<server>:<port>/webdav * /user:<user>
/persistent:no
```

<driveLetter> is the letter of the drive on which you want to mount the folder. You can use the wildcard "*" instead, so that Windows automatically chooses a free drive letter. <server> and <port> are the server name and port of the WebDAV server and <user> is your user name on this server. You will be prompted for the password.

Example

```
net use y: https://cms:8443/webdav * /user:michael /persistent:no
```

If you want to communicate via HTTP you have to use Digest authentication because Basic authentication is disabled in Windows Vista and Windows 7 when using HTTPS. Note that with Digest authentication you can not log in as LDAP user.

Troubleshooting

You get an error "System Error 67 has occurred. The network name cannot be found."

Possible cause

You have used Basic authentication with an HTTP connection.

Possible solution

Configure Basic authentication with an HTTPS connection or use Digest authentication as described in this chapter.

You get an error "The folder you entered does not appear to be valid. Please choose another."

Possible cause

You have used Basic authentication with an HTTP connection.

Possible solution

Remove `authentication.scheme.1=Basic` from `webdav.properties` or configure Basic authentication with an HTTPS connection.

Connecting to the Content Server

Configure the URL of the *CoreMedia Content Management Server* in the file `capclient.properties`. Set the property `cap.client.server.ior.url` to the value `http://<server>:<port>/coremedia/ior`.

Replace `<server>` with the name of the host where the server runs and `<port>` with its port. The port must match with the value of `cap.server.http.port` in the `contentserver.properties` file of the *Content Management Server*.

IBM WebSphere Application Server

This chapter describes the required adaptations to *CoreMedia WebDAV Support* to deploy it to IBM WebSphere Application Server. It assumes that you have read the common information on IBM WebSphere deployment which you can find in the *CoreMedia Operations Basics* manual.

In particular:

- Logging Configuration,
- Class Loader Configuration and
- ORB Configuration

Libraries

All libraries can be taken as is without exclusions or shared libraries.

ORB Configuration

To use the ORB of IBM WebSphere add the following entry to your `webdav.properties`:

```
filesystem.orb.jndi=java:comp/ORB
```

Have a look in the log file to see if your adaptation were successful:


```
2011-04-07 11:03:55 [ INFO] FilesystemImpl -  
  filesystem.orb.jndi: Using ORB com.ibm.CORBA.iiop.ORB@6f056f05  
  (server.startup : 0)
```

3.18 Troubleshooting

You start the server for the first time or after changes of the content type definition and an `ArrayOutOfBoundsException` occurs.

Possible cause:

The problem occurs when the number of content types is a multiple of 64 due to an error of the IBM XML parser used.

Possible solution:

Add an abstract dummy content type to the content type definition.

A publication is possible, but reports that no action was necessary to publish the content item. The content item, or the most recent version, does not appear on the Master Live Server.

Possible cause:

1. The properties - in particular the URL of the live server in `publisher.properties` - are incorrectly configured.

Possible solution:

1. Check the properties. It might point to the Content Server.
2. Restart the live server. The use of separate computers for the *Content ManagementServer* and *Master Live Servers* is recommended.

The server shows poor performance when accessing the database; in particular, it seems to be almost at a standstill every 24 hours.

Cause:

The database systems supported by *CoreMedia CMS* do not update their statistics data for tables and indexes automatically (except the MS SQL Server). This leads to the situation that database queries are not optimized and are therefore only processed inefficiently. Once every 24 hours, the *Content Server* places a query to release not referenced objects. If the statistics data and indexes are not optimized, this query cannot be processed optimally.

Solution:

Using database maintenance, update the statistics data daily for all tables and indexes of the corresponding CoreMedia database user on the production and live systems.

With an Oracle database the following SQL command must be executed, for example with `sqlplus` or another suitable program for each CoreMedia database user for the purpose of optimization:

```
call dbms_utility.analyze_schema('<Database user>', 'COMPUTE')
```

When using a DB2 database, optimization can be done with the corresponding menu entry in the DB2 Control Center.

The commands given should be executed either under the corresponding CoreMedia database accounts or an administration account with the appropriate privileges.

Note: The standard interval of 24 hours can be modified, if necessary, with the parameter `sql.store.collector.delay=<sec>`, in order to set a larger or smaller time span for memory checking. However, this should usually not be necessary.

The server no longer works correctly with an Oracle database and reports the following SQL Exception:

ORA-01000 maximum open cursors exceeded

Cause:

The configuration of the database is incorrect.

Solution:

The database administrator must adjust the Oracle configuration (see section 5.1.1.3.1) and restart the database.

You are using Oracle Solaris and while starting the server the error message "couldn't set locale correctly" appears.

Possible cause:

The package `SUNWceus` is not installed on the machine.

Possible solution:

Install the package.

4. Developing a Content Type Model

CoreMedia CMS manages content organized in freely configurable so-called content items. Content items normally contain the information of one entity. They may contain only a single information, an image, for example or may merge all information concerning a content object.

To have a consistent appearance and to avoid unnecessary effort, templates should be used for the creation of content items. *CoreMedia* supports this in the form of content types, which can be designed following object-oriented principles.

Content items in *CoreMedia CMS* are described by so-called *document properties* (also called *fields* or *properties* for short). An item of the type "press statement" for example may consist of properties such as:

- author
- date
- title
- summary
- textual content
- accompanying images

On the other hand, an image item has different content fields, such as dimensions and graphical data.

The more structured a content item is in specific fields, the more flexible is the access to contents from output templates, for web page construction, for example.

Furthermore, the *Site Manager* allows you to use content fields as search criteria. So it also pays off here to put some more effort in the creation of content types and to have as many properties as possible.

Since content type and source format of the content items are application-specific, *CoreMedia CMS* content types are not rigidly programmed. You have the possibility to design the content types and fields that you need to represent the maximum

amount of structured information from your content in *CoreMedia CMS*. In general, this design process will be done in cooperation with members of the editorial staff.

The content types can be defined in one or several XML files, to support a more modular content type assembly. Its structure is described more detailed in the following sections. In [Chapter 5, Appendix \[226\]](#) you learn how to install your content type file.

4.1 Properties

Each content type needs a specification of all properties that the corresponding content items have. The properties, presented as fields to the editor, vary, for example simple strings (such as for the author), XML for the textual content or binary data for graphics. This is determined in the content definition type with appropriate property types.

A content type may contain a maximum of 91 properties.



All field types have the attribute *Name*, defining the name by which the field can be referenced in an item.

The name can have a maximum length of 18 characters (`DateProperty` field names only 15 characters) and must not contain umlauts or other special characters ("§", "&", ...). Two fields in a content type different only in upper or lower case are not allowed. Furthermore, a name's last character must not be an underscore, since these field names are reserved for CoreMedia.



The property types are:

- `IntProperty`
- `StringProperty`
- `DateProperty`
- `XmlProperty`
- `BlobProperty`
- `LinkListProperty`

When defining XML elements and attributes in the following sections, the default namespace is assumed to be `http://www.coremedia.com/2008/documenttypes` and the namespace prefix `extensions` is assumed to be bound to the namespace URL `http://www.coremedia.com/2013/documenttypes-extensions`.

IntProperty

IntProperty

With an `IntProperty`, a field's value must be a whole number.

Example for creating an `IntProperty`:

```
<IntProperty Name="Priority"/>
```

StringProperty

StringProperty

In a `StringProperty` field, you can store unformatted text.

Table 4.1. Attributes of a StringProperty field

Property	Description
<code>Length</code>	Maximum number of characters that can be entered
<code>Utf8Length</code>	Maximum number of bytes. This attribute is optional. If you do not use this attribute, the value of the attribute <code>Utf8Weight</code> of the <code><DocTypes></code> element will be used to calculate the maximum byte length ($=Length * Utf8Weight / 100$). If you set none of the attributes <code>Utf8Length</code> and <code>Utf8Weight</code> , <code>Utf8Weight=300</code> will be used (see below for details) to define the byte length.
<code>extensions.translatable</code>	If set to true, this <code>StringProperty</code> field is exported to an XLIFF file during translation.
<code>extensions.automerge</code>	If set to true, the contents of this property field are merged automatically from the master content if the translation task is accepted by the owner of the derived content.

Example for creating a `StringProperty`:

```
<StringProperty Name="Title" Length="200" Utf8Length="500"/>
```

The maximum number of characters to be entered is limited to 200 and the maximum length would be 500 bytes.

DateProperty

DateProperty

Such fields display dates and times. Editors for simple input are available

Example for creating a `DateProperty`:

```
<DateProperty Name="Date"/>
```

XmlProperty

XmlProperty

Whereas for some properties, such as the author, a simple character string is sufficient, an `XMLProperty` field specifies an XML document field. `XmlProperty` fields require a `Grammar` attribute which refers to the DTD or Schema of the content.

Table 4.2. Attributes of an XmlProperty field

Property	Description
<code>Grammar</code>	Defines the DTD or Schema that should be used for this content. <code>coremedia-richtext-1.0</code> is the standard grammar for

Property	Description
	CoreMedia RichText fields. The special predefined grammar <code>coremedia-struct-2008</code> is used for properties that contain structured objects at the level of the <i>Unified API</i> , but that are represented as markup internally. Structs are typically used for highly dynamic configuration tasks.
<code>extensions.translatable</code>	If set to true, this property field is exported to an XLIFF file during translation.
<code>extensions.automerge</code>	If set to true, the contents of this property field are merged automatically from the master content if the translation task is accepted by the owner of the derived content.

Example for creating an `XmlProperty` with Structs:

```
<XmlProperty Name="Config" Grammar="coremedia-struct-2008"/>
```

BlobProperty

BlobProperty

You can use a `BlobProperty` field to store binary data in *CoreMedia Digital Experience Platform 8*. The attribute `MimeType` defines which binary types you can store in the field.

Example for creating a `BlobProperty`:

```
<BlobProperty Name="Graphic" MimeType="image/*"/>
```

In this property you can store images of all types, such as PNG, JPEG, GIF.

LinkListProperty

LinkListProperty

Some content items belong together in terms of content, even if they do not reference each other with internal links in XML texts. For example, images belong to a press report which are not explicitly mentioned in the text as illustrations. The number of images differs from case to case, so that it is not practical to define one field per image in the content type. For such cases there is the `LinkListProperty`, which holds a list of content items.

Table 4.3. Attributes of *LinkListProperty*

Name	Description
<code>LinkType</code>	If the attribute <code>LinkType</code> is given, all content items of the list must be of the same content type as determined with that attribute or a subtype of that content type.
<code>Min</code>	The minimum amount of references for this link property.

Name	Description
Max	The maximum amount of references for this link property.
extensions:weak-Link	<p>Target items that are linked via weak link property will not be published or withdrawn together with the linking content item automatically. The default setting is "false".</p> <p>At Unified API level non existing link targets are represented by a destroyed content object. During content bean creation the <code>ContentBeanFactory</code> converts these destroyed contents to <code>null</code> and filters them from created content bean lists.</p> <p>Caution!</p> <p>The introduction of weak links to a document type model comes with a caveat:</p> <ul style="list-style-type: none"> → Weak links can cause dead links in the live environment. → Unified API client code has to cope with destroyed contents. → Content bean code has to cope with <code>null</code> values and filtered lists of content beans.

Example for creating a `LinkListProperty`:

```
<LinkListProperty Name="Images" LinkType="Image"/>
```

Only content items of type `Image` or subtypes are allowed in this link list.

Indexing

Indexing

If you initialize the *Content Server*, a lot of indices are created in the database. They will speed up the operation of the server, loading of resources, for instance. However, no indices are generated for the user defined content properties by default, because:

- Indices for properties would only speed up queries.
- Indices for properties would slow down creation and changing of content.

If you need to have a very fast query via the Site Manager or the Query-API (but not the *CoreMedia Search Engine*) you might create indices for the `DateProperty`, `IntProperty` and `StringProperty`.

It's possible to create these indices automatically, if you set the attribute `Index` of the `DocType`, `DateProperty`, `IntProperty` or `StringProperty` element

to "true". If you set `Index="true"` for the `DocType` element, indices would be created for all `DateProperty`, `IntProperty` and `StringProperty` fields contained in this content type.

Translatable Properties

In order to support automatic translation processes, properties of the document model can be marked as translatable. When translating a document automatically, only properties that set the attribute `extensions:translatable="true"` should be included.

```
<StringProperty Name="keywords" Length="1024"
  extensions:translatable="true"/>
```

In general, most string and richtext properties will be marked translatable, excluding those that contain technical strings and identifiers.

Automatically Merged Properties

Usually all non-translatable properties in the master content will be applied automatically to the derived content when a translation task is accepted. This helps to keep binary and structural data in sync between sites, such as images, crops, settings, and the navigation hierarchy, and complements the XLIFF-based update of translatable properties. To enable the automatic merge for a translatable property or disable the automatic merge for a non-translatable property the `extensions:automerge` attribute has to be attached.

```
<XmlProperty Name="settings" Grammar="coremedia-struct-2008"
  extensions:translatable="true" extensions:automerge="true"/>
```

String Properties and UTF-8 Encoded Database

In an UTF-8 encoded database, up to three bytes are required to store a single character. ASCII characters are stored in one byte and non-ASCII Latin-1 characters (such as German umlauts) in two bytes. Asian characters need three bytes for representation.

The attribute `Length` in the element `StringProperty` defines the maximum length of allowed characters. The actual byte size of the generated `varchar` table column in the UTF-8 encoded relational database is a product of the value of `Length` with an additional factor.

To avoid representation problems and to store even Asian characters the simplest but wasteful way would be to multiply the string `Length` defined in the `StringProperty` element with a factor of three to compute the length of the generated `varchar` table column that holds the string values. In addition to the waste of storage, this could lead to database problems because database boundaries are

reached when many string properties with big `Length` values are specified in one content type or within a content type inheritance hierarchy. To be more flexible, every `StringProperty` element can have an attribute `UTF8Length` to specify the maximum number of bytes to hold a string property value. The value must be at least equal and at most three times greater than the value of the `Length` attribute. If for nearly all string properties in the content type schema the ratio of maximum character length to byte length is the same, one can simply add an attribute `Utf8Weight` to the `DocumentTypeModel` element. The value must be between 100 and 300 and, when divided by 100, defines the ratio between character and byte length for all string properties in the type schema where the attribute `Utf8Length` is not explicitly set. The default value for the `Utf8Weight` attribute, if omitted, is 300.

For example the value of the attribute `Utf8Weight` can be set to 130. If a string property with a given `Length` attribute value of 50 and without `Utf8Length` attribute is specified, then the maximum byte length is $130/100 * 50 = 65$. This may be enough for German text strings, because a string with a character length of 50 could contain 15 two byte encoded characters (such as German umlauts). If the length of a string is less than 49 characters, then even more non-ASCII characters are allowed.

If you are using a Microsoft SQL Server, things are easier. The CoreMedia server uses `nvarchar` for string properties in the SQL Server which can simply be used for Unicode strings. So, you only need to set the `UTF8Weight` attribute of the `DocTypes` element to 100. No triple size is needed, no `UTF8Length` attributes for string properties are required.



4.2 Creating Content Type Definitions

In *CoreMedia CMS* you define your content type definitions in XML files. *CoreMedia CMS* allows you to administrate content type definitions modularly. That means, that content types can be defined in different independent files, instead of one monolithic file. The *Content Server* loads and combines all content type files which match a given pattern. You can define the pattern in the `cap.server.documentTypes` property in the `contentserver.properties` file (see [Section 5.1, "Configuration in contentserver.properties"](#) [227] for details).

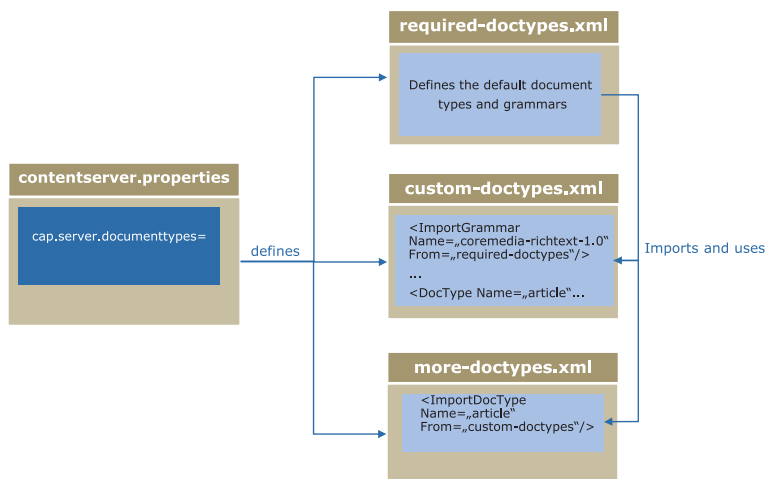


Figure 4.1. Schema of content type definitions

You can also link from one content type file to content types and grammars defined in other files.

Examples:

In this example, the content type `BaseDocument` and the grammar `mycompany-richtext` are defined in the `base-doctypes.xml` file and are also used in the `specific-doctypes.xml` file. To this end, both are imported using the `<ImportDocType>` and `<ImportGrammar>` elements.

```

<DocumentTypeModel
  xmlns="http://www.coremedia.com/2009/documenttypes"
  xmlns:extensions="http://www.coremedia.com/2013/documenttypes-extensions"
  Name="Base-Doctypes">

```

Example 4.1. The `base-doctypes.xml` file

```

<XmlGrammar Name="mycompany-richtext" Root="company"
  PublicId="-//MyCompany//DTD Richtext 1.0//EN"
  SystemId="lib/xml/mycompany-richtext.dtd"/>

<DocType Name="BaseDocument" Abstract="true">
  <StringProperty Name="Titel" Length="200"/>
</DocType>

</DocumentTypeModel>
    
```

Example 4.2. The specific-doctypes.xml file

```

<DocumentTypeModel
  xmlns="http://www.coremedia.com/2009/documenttypes"
  xmlns:extensions="http://www.coremedia.com/2013/documenttypes-extensions">

  <ImportDocType Name="BaseDocument" From="Base-Doctypes"/>
  <ImportGrammar Name="mycompany-richtext" From="Base-Doctypes"/>

  <DocType Name="Text" Parent="BaseDocument">
    <XmlProperty Name="Text" Grammar="mycompany-richtext"/>
  </DocType>

</DocumentTypeModel>
    
```

4.2.1 Structure of Content Type Definitions

The basis for your content items in a *CoreMedia CMS* system is the XML file `config/contentserver/doctypes/sample-doctypes.xml`. It contains the root element `<DocumentTypeModel>` which serves as a container for the `<DocType>`, `<XmlGrammar>` and `<XmlSchema>` elements. It imports the grammar for CoreMedia rich text from the `required-doctypes.xml` file which is located in the `framework/doctypes` directory in `cap-server.jar`. The `required-doctypes.xml` contains the definitions which are essential for *CoreMedia CMS*. The next example shows a simple content type definition.

```

<ImportGrammar Name="coremedia-richtext-1.0"
  From="required-doctypes"/>

<DocType Name="article">
  <StringProperty Name="Author" Length="200"/>
  <StringProperty Name="Headline" Length="200"/>
  <DateProperty Name="Date"/>
  <XmlProperty Name="Summary"
  Grammar="coremedia-richtext-1.0"/>
  <LinkListProperty Name="Images" LinkType="Image"/>
</DocType>

<DocType Name="Image">
  <IntProperty Name="Width"/>
  <IntProperty Name="Height"/>
  <BlobProperty Name="Image" MimeType="image/*"/>
</DocType>
    
```

Example 4.3. Example of a content type definition

Content type names

Every content type has a unique name, specified by the attribute *Name*. This name will be used by the *Content Management Server* to define unique data base identifiers like table names, primary key constraint names, foreign key constraint names and index names. The name has a maximum length of 11 characters, is not case-sensitive and can contain letters, numbers, underscore and dash but no umlauts or other special characters ("§", "&", ...). For a localization of content type and property names see the [Site Manager Developer Manual] and the [CoreMedia Studio Developer Manual].

You are not allowed to use the following words for content type names:

- Names of SQL commands.
- Names which are reserved for CoreMedia as database table names:
BlobCodeTable, BlobData, Blobs,ChangeLog, CmGroups, CmLicenses, CmProcessorUsage,CmRules, CmUserGroup, CmUsers, Dictionary, Editor-Preferences, FolderIndex, LinkChangeLog, LinkLists, MaxIds, MaxMemberIds, MySQLBlobStore, Preferences, Query, QueryIdTable, Replicator-IdTable, Resources, SgmlData, SgmlGrammar, SgmlText, System, Texts, TrashIdTable, WfDefinitions, WfMaxDeletedId, WfPendingSignals, WfProcessInstances, WfProcesses, WfTaskInstances, WfTasks, WfVariables
- Names corresponding to the name scheme `<doctype><digit><digit>"i"<char>*`, where `<doctype>` stands for an already defined content type name, `<digit>` for a number and `i<char>*` for a word which begins with the letter "i".



If you have used content type names which exceeded the recommended length of 11 characters (which are longer than 15 characters to be exact) and you switch to another database you might exceed database limitations. This is especially true for IBM DB2. In order to circumvent these problems there exists three attributes for the `DocumentType` element.

- `PkName`: This attribute defines a name for the primary key constraint of the content type table.
- `ResourceFkName`: This attribute defines a name for the foreign key constraint from the content type specific tables.
- `IndexName`: This attribute defines the index name for columns of content type specific tables.

Example:

Your content type name is "ProductAttributes" (`<DocumentType name="ProductAttributes">`). This name contains 17 characters. The *CoreMedia Server* will

extend this name by up to three characters to obtain unique database identifiers. Unfortunately, the DB2 database has a limitation of 18 characters which will be exceeded by these names. In order to use your content type name furthermore you define unique names for the database using the attributes described above. Your new content type definition will look similar to the following example:

```
<DocumentType name="ProductAttributes" PkName="pk_ProdAttr"
ResourceFkName="fk_ProdAttr" IndexName="i_ProdAttr">
```

Three content types, *Dictionary*, *Query* and *Preferences*, are already defined in the `required-doctypes.xml` file. All types are essential for some components of *CoreMedia CMS* and must therefore neither be renamed nor deleted.

Defining the grammar

The `<XmlGrammar>` element with the attributes `Name`, `Root`, `Parent` and `SystemId` is used to refer to DTDs from `XMLProperty` elements. The following example defines the *CoreMedia* rich text grammar:

```
<XmlGrammar Name="coremedia-richtext-1.0"
  Root="div"
  PublicId="//CoreMedia//DTD Rich Text 1.0//EN"
  SystemId="classpath:xml/coremedia-richtext-1.0.dtd"/>
```

Table 4.4. Attributes of `XmlGrammar` element

Property	Description
Name	The grammar name which you can use to reference the DTD
Root	The name of the XML root element
PublicId	The public identifier of the XML grammar
SystemId	A path that is either absolute or relative to <code>WEB-INF</code> which specifies the location of the XML grammar in the <i>CoreMedia CMS</i> server's file system or classpath. The <code>SystemId</code> can also be a URL from where the <i>CoreMedia CMS</i> server loads the XML grammar on startup.
Parent	<i>CoreMedia CMS</i> has an inheritance concept for XML grammars. If this attribute is set to the name of another XML grammar, the XML grammar may be used when overriding a property with the other XML grammar.

The following `<XmlGrammar>` elements are predefined:

- `coremedia-richtext-1.0`
- `coremedia-dictionary`
- `coremedia-query`

→ coremedia-struct-2008

→ coremedia-preferences

The latter four belong to the Dictionary, Query, EditorPreferences, and Preferences content types respectively. The grammar coremedia-struct-2008 is also used frequently in *CoreMedia Blueprint*.

If you want to use XML Schemas instead of DTDs you can specify the `<XmlSchema>` element with the attributes `Name`, `SchemaLocation`, `Language` and `Parent`. The following example defines the grammar `custom-schema` with a schema file `custom.xsd` located in the package `my.pkg`, which is available in a JAR file deployed with the *Content Server* and any other server which needs to validate the XML as for example the *Workflow Server* if actions access properties using this grammar:

```
<XmlSchema Name="custom-schema"
  SchemaLocation="classpath:my/pkg/custom.xsd
  http://www.w3.org/1999/xlink.xsd"
  Language="http://www.w3.org/2001/XMLSchema">
```

Example 4.4. Using XML Schemas

Property	Description
Name	The name which you can use to reference the Schema
SchemaLocation	A path that is either absolute or relative to WEB-INF which specifies the location of the XML schema in the <i>Content Server's</i> file system or classpath. The <code>SchemaLocation</code> can also be a URL from where the <i>CoreMedia CMS</i> server loads the XML schema on startup. You can define multiple schemas, separated by white spaces.
Language	For example <code>http://www.w3.org/2001/XMLSchema</code>
Parent	Parent: <i>CoreMedia CMS</i> has an inheritance concept for XML schemas. If this attribute is set to the name of another XML schema, the XML schema may be used when overriding a property with the other XML schema.

Table 4.5. Attributes of the XMLSchema element

You specify a DTD grammar or XML schema once in the content type declaration and can refer to it with the attribute `Grammar` in the `XmlProperty` element:

```
<DocType Name="Article">
  <StringProperty Name="Headline" Length="200"/>
  <XmlProperty Name="Text"
    Grammar="coremedia-richtext-1.0"/>
  <XmlProperty Name="Summary"
    Grammar="coremedia-richtext-1.0"/>
  <XmlProperty Name="Comment">
```



```
Grammar="custom-schema"/>
</DocType>
```

Importing grammars and document types

You can import a grammar or schema into your content type file using the `<ImportGrammar>` element. The following example imports the "coremedia-richtext-1.0" grammar defined in the `required-doctypes.xml` file:

```
<ImportGrammar Name="coremedia-richtext-1.0"
From="required-doctypes"/>
```

Property	Description
Name	The name of the grammar or schema you want to import.
From	The name of the XML file, where the grammar or schema is defined (without the file extension).

Table 4.6. Attributes of the `ImportGrammar` element

You can also import content types that are defined in different content type files. This is necessary, if you want to use this content type as a parent or as a target of a `LinkList` property. The following example would import the type `Article` from the `editorial-types.xml` file.

```
<ImportDocType Name="Article"
From="editorial-types"/>
```

The attributes have the same meaning as for the `ImportGrammar` element.

4.2.2 Inheriting Content Types

CoreMedia CMS has an inheritance concept for content types and properties. In the `<DocType>` element, there is an optional attribute, `Parent`. If this attribute is set to the name of another content type, the new content type inherits all fields of its parent content. A property in the inheriting content item with the same name and type as a property in the parent item will override the parent's property. You have to set the attribute `Override` to `true` in a `Property` element to use overriding. Overriding is only possible if it specializes the type of the property.

- Strings of the inheriting content type have to be shorter or equal.
- blobs must use a more specific MIME type.
- XML grammars must be a child of the parents XML grammar.
- For link lists, the link type must be a subtype of the parent's link type or the parent must be untyped.

In addition, a content type can be defined as abstract. For it, the attribute `Abstract` must be set to "true" in the `<DocType>` element. By default, a content type is defined as not abstract. A content of abstract type cannot be created by the server.

The following example shows the content type **Image** and a special type, **ImageWithThumbnail**, which offers the field **Thumbnail** in addition for a reduced version of the image.

```
<DocType Name="Image">
  <IntProperty Name="Width"/>
  <IntProperty Name="Height"/>
  <BlobProperty Name="Image" MimeType="image/*"/>
</DocType>

<DocType Name="ImageWithThumbnail" Parent="Image">
  <BlobProperty Name="Thumbnail" MimeType="image/*"/>
</DocType>
```

Example 4.5. Example for content type inheritance

In this way, content items offer the usual flexibility of the object-oriented world: according to the context, you can allow general or special content types in link lists or templates.

For good system performance, however, you should not inflate the inheritance tree with too many generic intermediate types, but limit this to types which will actually be required in the foreseeable future.

The order of the type definitions is relevant to inheritance: a content type used as the parent type must be defined previously. In this way, cyclic construction of inheritance relationships is ruled out.

4.2.3 Attaching Properties to Existing Content Types

In *CoreMedia CMS* it is possible to define multiple content type definitions in separate files (see [Section 4.2, “Creating Content Type Definitions” \[208\]](#)). At startup time the *Content Server* merges these files to create the complete content type hierarchy. You can use these files to alter an existing content type definition without changing the base definition file in three ways:

- Add new content types (see [Section 4.2.1, “Structure of Content Type Definitions” \[209\]](#)).
- Inherit from existing content types (see [Section 4.2, “Creating Content Type Definitions” \[208\]](#)).
- Attach new properties to existing content type definitions.

The first two ways are described in [Section 4.2.1, “Structure of Content Type Definitions” \[209\]](#) and [Section 4.2, “Creating Content Type Definitions” \[208\]](#). In this section you will learn how to add properties to an existing content type.

In order to add properties to an existing content type a `DocTypeAspect` element has been introduced, that allows you to attach properties to a content type specified by the `TargetType` attribute. More than one `DocTypeAspects` might attach properties to the same target `DocType`, but you have to take care that no property name clashes occur. The relationship between target and aspect is therefore of the cardinality 1:n. As a limitation to ensure consistent behavior in subtypes, it is neither allowed to alter existing or inherited properties nor attach properties that are already defined in subtypes of the `TargetType`. To define `DocTypeAspects` the definition file must be valid against the `coremedia-doctypes-2009.xsd` schema.

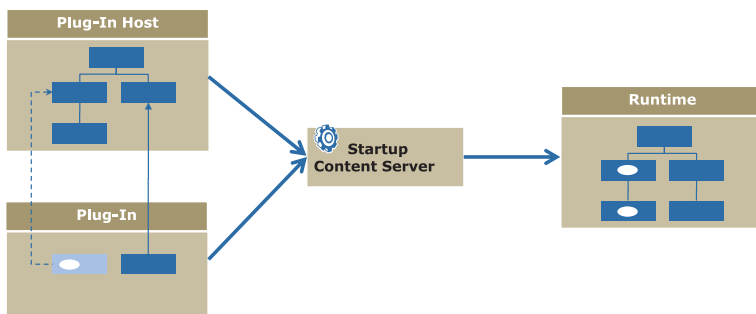


Figure 4.2. Extending content types

Example

An abstract content type `CMMedia` is defined in file A. `CMMedia` is an abstract content type and is the root of a media document type hierarchy.

```
<DocType Abstract="true" Name="CMMedia" Parent="CMTeasable">
  <LinkListProperty Name="master" Max="1" LinkType="CMMedia"
    Override="true"/>
  <XmlProperty Name="caption"
    Grammar="coremedia-richtext-1.0"/>
  <StringProperty Name="alt" Length="128"/>
</DocType>
```

The content type `CMMedia` is extended with a string property named "copyright" in file B. If you want to change a content type via `DocTypeAspect`, you have to import the content type before, using the `<ImportDocType>` element. Because of the `DocTypeAspect` semantics, the `copyright` property is inherited by all subtypes of `CMMedia` without changing their definitions.

```
<ImportDocType Name="CMMedia"/>
<DocTypeAspect TargetType="CMMedia">
  <StringProperty Name="copyright" Length="128"/>
</DocTypeAspect>
```

4.3 Schema Migration

In general, content types will be defined in the development phase of your content model. They are the "backbone" of your CoreMedia application and therefore are assumed to be stable. Nevertheless, new requirements might demand changes in the content model while preserving the existing content. In this chapter you will find a description on how to add, rename and remove content types and attributes to and from your content model.

Always make a backup of your data before schema migration. Stop the *Content Server* when you change content types. The changes take effect not until the *Content Server* has been started. Choose log level "Debug" and check the server log file for the impact of the new content types.



4.3.1 The Database Schema

In this section, you will find a description of the database schema used by the *CoreMedia CMS* system. The *Content Server* analyzes the content types file and automatically creates the database schema from it.

Resource tables

The following tables hold information concerning resources:

- `<documenttypename>`: This table has the name of the corresponding content type.
- `resources`: Holds information for each resource existing in the CoreMedia system.

Property tables

The following tables hold information concerning properties:

- `blobs`: Connects `blobdata` with the content type table.
- `blobdata`: Contains the actual blobs content in a `LONG RAW` column.
- `texts`: Connects `sgmldata` and `sgmltext` with the content type table.
- `sgmldata`: Contains markup data of an XML property.
- `sgmltext`: Contains actual text data of an XML property.
- `linklist`: Connects the content item which links with the item to which it links.

User Management

The following tables hold information for the user management:

- `cmusers`: Contains the user data.
- `cmgroups`: Contains the group data.
- `cmusergroup`: Connects users with groups.
- `cmrules`: Contains the rules of user groups.

The <documenttypename> table

The Content Server creates the following columns for each property of the described type. Properties which are inherited are defined in the parent content type table and in all inheriting content type tables. Therefore, you have to change the table of the parent type and the tables of each subtype if you change a property of the parent type.

- `IntProperty`: Column of type `NUMBER(10)` with the name of the property.
- `StringProperty`: Column of type `VARCHAR(<UTF-8 length>)`. The UTF-8 length results from the length defined via the `length` attribute of the `StringProperty` multiplied by an UTF-8 weight. By default, this factor is three. See [Section 4.1, "Properties" \[202\]](#) for a description of the UTF-8 length.
- `DateProperty`: Column of type `DATE` with the name of the property and another column of type `VARCHAR(30)` with the name `<PropertyName>_tz`.

In addition, each content type table contains the following system attributes:

Attribute	Description
ID_	Unique ID of the content item. Attribute is never null.
VERSION_	Version of the content item. Attribute is never null.
ISAPPROVED_	Marker if this version of the content item is approved ("1") or not ("0").
ISPUBLISHED_	Marker if this version of the content item is published ("1") or not ("0").
EDITORID_	User ID of the editor.
APPROVERID_	User ID of the approver.
PUBLISHERID_	User ID of the publisher.
EDITIONDATE_	Date of the last change.
APPROVALDATE_	Date of the approval.

Table 4.7. System attributes of the content type table

Attribute	Description
PUBLICATIONDATE_	Date of the publication

The primary key of the content type tables consists of the `ID_` and `VERSION_` columns. There exists a foreign key connection to the `resources` table via the `ID_` column.

The blobdata table

The primary key of the `blobdata` table consists of the following properties:

- `DocumentId`: ID of the content item which contains the blob.
- `DocumentVersion`: Version of the content item which contains the blob.
- `PropertyName`: Name of the property which contains the blob.

The linklists table

The primary key of the `linklists` table consists of the following properties:

- `SourceDocument`: ID of the content item which links to the other document.
- `SourceVersion`: Version of the content item which links to the other document.
- `PropertyName`: Name of the property which contains the link list.
- `LinkIndex`: Position of the link in the link list.

4.3.2 Adding Content Types

Adding content types is the simplest task. Proceed as follows:

1. Add the new content types to the content types definition file of the *Content Server*.
2. Optional create a secondary file containing new content type definitions and add it to `cap.server.documentTypes` in `contentserver.properties`.
3. Restart the *Content Server*.

The *Content Server* automatically detects the new content types and creates the necessary database tables. The *Content Server* creates the new tables in the default table space of the CoreMedia database user. If you have distributed the CoreMedia tables on different table spaces, you have to assign the new tables manually to the appropriate table space.

4.3.3 Renaming Content Types

The *Content Server* does not support automatic renaming of content types, as it is hard to detect this situation without additional operator assistance. In the following, you find a description to consistently rename content types in the `doctypes.xml` file and in the database.

1. Make sure that all *Replication Servers* (and other repository listeners) are idle, that is, that there are no pending events in the *Content Server's* `ChangeLog`. Otherwise, events might refer to an incorrect content type name later.
2. Stop the *Content Management Server* and all *Live Servers*.
3. Rename the content type in the content type definition file, including the `Parent` attribute of all subtypes and the `Target` attribute of all `LinkLists`.
4. Rename the database table that represents the content type to the new name. Depending on the database, this is possible using either an `ALTER TABLE` statement, or by creating a new table, copying all data, and deleting the old table.
5. Rename occurrences of the content type name in the `Resources` table:

```
UPDATE Resources SET documentType_ = 'myNewName' WHERE
documentType_ = 'myOldName'
```

6. Rename occurrences of the content type name in the `Rules` table:

```
UPDATE CmRules SET docType = 'myNewName' WHERE docType =
'myOldName'
```

7. Start the *Content Servers*.

Note that you may have to adapt various parts of your application, such as `Queries` stored in the repository, template file names and `EL` expressions in templates, generated `ContentBeans` etc.

If point 1 (idle repository `ChangeLog`) cannot be ensured, you can proceed as follows:

1. Truncate the `ChangeLog` as far as possible, as described in [Section 3.12, "Truncate the ChangeLog" \[75\]](#).
2. Update all `ChangeLog` entries that refer to content type names (the column depends on the event code):

```
UPDATE changelog SET s2='myNewName' WHERE s2='myOldName' AND i1
IN (1, 2);
```

```
UPDATE changelog SET s1='myNewName' WHERE s1='myOldName' AND i1
IN (36,37);
```

3. After restarting the *Content Server*, all clients connecting to this server will only see the new content type name in content repository events.

4.3.4 Deleting Content Types

Before you delete content types, make sure that the data is no longer needed.



Delete content type

Deletion of content types is best done with a mixture of server utilities and SQL commands. Proceed as follows:

1. Start a query from the *Site Manager* to select all content items of the content type to delete.
2. Mark all these content items for deletion.
3. Publish all the content items.
4. Use the `cleanrecyclebin` utility to remove all the content items from the recycle bin.
5. Stop the *Content Management Server* and the *Live Servers*.
6. Remove the content types to delete from the content types definition files of all *Content Servers*.
7. Verify that the table which represents the content type is empty.
8. Drop the table which represents the content type from the databases.
9. Delete all rules that are attached to the document type by using the *Site Manager* or by settling the following SQL statement:

```
DELETE FROM CmRules WHERE doctype = '<docTypeName>';
```

If the rules are deleted by using SQL, all servers and clients need to be restarted, as the document type is being cached.

10. Start the *Content Servers* again.

4.3.5 Adding Properties

How to add properties depends on the property type.

For properties of the type,

→ `XmlProperty`

→ `BlobProperty`

→ `LinkListProperty`

proceed as follows:

1. Add the new property types to the content types definition file.
2. Restart the *Content Server*.

For properties of all other types proceed as follows:

1. Stop the *Content Server*.
2. Change the content types definition file.
3. Be sure, that the properties `sql.schema.checkColumns` and `sql.schema.alterTable` in the `<ContentServerHome>/properties/corem/sql.properties` file are set to "true"
4. Restart the *Content Server*.

4.3.6 Renaming Properties

The *Content Server* does not support automatic renaming of properties, as it is hard to detect this situation without additional operator assistance. In the following, you find the steps necessary to consistently rename properties in the `doc types.xml` file and in the database.

Generally, all *Content Servers* should be migrated at the same time, because an attempt to replicate or publish content between *Content Servers* with different schema will fail. Alternatively, it should be made certain that a *Replication Live Server* cannot connect to a *Master Live Server* while their schemata differ, or that a *Content Management Server* cannot publish to a *Master Live Server* while their schemata differ.

Since property names do not occur in the repository `ChangeLog`, it is possible to migrate the *Servers* even when the *Content Server* and repository listeners are not idle before shutdown.

For properties of the type `XmlProperty`, proceed as follows:

1. Stop the *Content Server*.

2. Rename the property in the content types definition file.
3. Execute the following SQL statement with the declaring content type and all its subtypes:

```
UPDATE Texts SET propertyName = 'myNewName'
WHERE propertyName='myOldName' AND EXISTS
(SELECT * FROM Resources WHERE id=documentId
AND documentType IN
('myType', 'mySubtype1', 'mySubtype2',...));
```

4. Rename the internal links. Execute the following SQL statement with the declaring content type and all its subtypes:

```
UPDATE LinkLists SET propertyName = 'myNewName'
WHERE propertyName='myOldName' AND EXISTS
(SELECT * FROM Resources WHERE id=documentId
AND documentType IN
('myType', 'mySubtype1', 'mySubtype2',...));
```

5. Start the *Content Server*.

For properties of the type `BlobProperty`, proceed as follows:

If inline images in XmlMarkup contain the property name of the referenced image blob, CoreMedia recommends not to change the property name. You would need to program a Unified API client to change the current versions of the XmlMarkup properties and there is no trivial way to change older versions.



1. Stop the *Content Server*.
2. Rename the property in the content types definition file.
3. Execute the following SQL statement with the declaring content type and all its subtypes:

```
UPDATE Blobs SET propertyName = 'myNewName'
WHERE propertyName='myOldName' AND EXISTS
(SELECT * FROM Resources WHERE id=documentId
AND documentType IN
('myType', 'mySubtype1', 'mySubtype2',...));
```

4. Start the *Content Server*.

For properties of the type `LinkListProperty`, proceed as follows:

1. Stop the *Content Server*.
2. Rename the property in the content types definition file.
3. Execute the following SQL statement with the declaring content type and all its subtypes:

```
UPDATE LinkLists SET propertyName = 'myNewName'
WHERE propertyName='myOldName' AND EXISTS
(SELECT * FROM Resources WHERE id=sourceDocument
AND documentType IN
('myType', 'mySubtype1', 'mySubtype2',...))
```

4. Start the *Content Server*.

For properties of the type `IntProperty` or `StringProperty`, proceed as follows:

1. Stop the *Content Server*.
2. Rename the property in the content types definition file.
3. Rename the database column representing the property in the database tables for the declaring document type and all its subtypes. Depending on the database, this is possible either
 - a. using an `ALTER TABLE` statement renaming the column, or
 - b. by adding a new column using an `ALTER TABLE` statement, copying the data from the old to the new column, and setting the old column to `NULL` (for DB2, for example), or
 - c. by creating a new table, copying all data, and deleting the old table.
4. Start the *Content Server*.

For properties of the type `DateProperty`, proceed as follows:

1. Stop the *Content Server*.
2. Rename the property in the content types definition file.
3. Rename the database column representing the property in the database tables for the declaring document type and all its subtypes, as described in the `IntProperty / StringProperty` case above.
4. Rename the database column storing the property's time zone, in the database tables representing the declaring document type and all its subtypes. The column name is the property name suffixed with "`_tz`".
5. Start the *Content Server*.

4.3.7 Deleting Properties

Before you delete properties, make sure that the data is no longer needed.



In order to delete properties from content types proceed as follows:

1. Stop the *Content Servers*.

2. Remove the attributes from the content types definition file.
3. Delete the property data from the database as described in the following table.

Property type	How to change
IntProperty DateProperty StringProperty	Remove the column representing the property from the database tables for the declaring document type and all its subtypes. For example: Remove the attribute <code>Source</code> of type <code>StringProperty</code> from the document type <code>Article</code> using the following SQL statement: <pre>ALTER TABLE Article DROP column "Source"</pre>
BlobProperty	Remove the property information from the <code>blobs</code> table. Don't delete anything from <code>blobdata</code> , this will be done by the <i>Content Server</i> ! Attribute names are only unique for one content type. Therefore, you have to check if the property you want to delete belongs to the proper content type. For example: Delete the <code>BlobProperty</code> named <code>Logo</code> from the content type <code>Article</code> . <pre>DELETE FROM Blobs WHERE propertyname = 'Logo' AND documentid = (SELECT DISTINCT id_ FROM Article WHERE id_ = documentid); COMMIT;</pre>
XmlProperty	Remove the property information from the <code>texts</code> table and the links contained in the <code>XmlProperty</code> from the <code>linklists</code> table. Don't delete anything from <code>sgmldata</code> or <code>sgmltext</code> , this will be done by the <i>Content Server</i> ! Attribute names are only unique for one content type. Therefore, you have to check if the property you want to delete belongs to the proper content type. For example: Delete the <code>XmlProperty</code> named <code>Text</code> from the content type <code>Article</code> . <pre>DELETE FROM Texts WHERE propertyname = 'Text' AND documentid = (SELECT DISTINCT id_ FROM Article WHERE id_ = documentid); COMMIT;</pre> <pre>DELETE FROM LinkLists WHERE propertyname = 'Text' AND sourceDocument = (SELECT DIS TINCT id_ FROM Article WHERE id_ = source Document); COMMIT;</pre>

Table 4.8. How to delete properties of different type

Property type	How to change
LinkListProperty	<p>Remove the property information from the linklists table. Attribute names are only unique for one content type. Therefore, you have to check if the property you want to delete belongs to the proper content type.</p> <p>For example: Delete the LinkListProperty named RelatedDocuments from the content type Article.</p> <pre>DELETE FROM LinkLists WHERE propertyname = 'RelatedDocuments' AND sourceDocument = (SELECT DISTINCT id_ FROM Article WHERE id_ = sourceDocument);</pre> <p>COMMIT;</p>

5. Appendix

This chapter describes the content of all configuration files which you can use to configure the *CoreMedia Content Servers*.

Some property files contain additional property entries which are not described in the Manual. As a rule, these properties are for special, system-relevant settings which you should not change.

5.1 Configuration in `contentserver.properties`

Most of the *Content Server* configuration is done in `contentserver.properties`.

In addition, the following setting in `contentserver.properties` should be in sync with the corresponding setting in `server.xml`:

→ `cap.server.http.port`

See the description of the properties for more details. The corresponding settings in the `server.xml` file of the servlet container are shown in XPath syntax.

Property	Value	Default	Description
<code>cap.server.http.port</code>	Port		This property defines the HTTP(S) port of the application container containing the <i>Content Server</i> . The entry <code>/Server/Service/Connector@port</code> in <code>server.xml</code> has to have the same value as this property.
<code>cap.server.ORBServerHost</code>	String		The host of the ORB which the <i>Content Server</i> instantiates. This property is deprecated and is ignored when using the application server's ORB.
<code>cap.server.ORBServerPort</code>	Int		The port of the ORB which the <i>Content Server</i> instantiates. This property is deprecated and is ignored when using the application server's ORB.
<code>cap.server.ORBServerSSLPort</code>	Int	0	The port of the ORB which the <i>Content Server</i> instantiates for SSL encrypted communication. Set to 0 or leave empty if you do not want an SSL server socket. This property is ignored when using the application server's ORB.
<code>cap.server.repository.home</code>	Path	/Home	Using this property, you can define the folder which will be used to store the home folders of the users. The whole folder hierarchy of the home folders is only visible to the administrator. Other user will only see one home folder

Table 5.1. *contentserver.properties*

Property	Value	Default	Description
			with the path defined in <code>cap.server.repository.home</code> containing his personal files, such as the preferences. The default folder is <code>/Home</code> which will be automatically created by the system. If you define another folder, you need to create this folder by your own.
<code>cap.server.repository.system</code>	Path	<code>/System</code>	Using this property, you can define the system folder. It contains for example the public dictionary of the spell checker. The default folder is <code>/System</code> which will be automatically created by the system. If you define another folder, you need to create this folder by your own.
<code>cap.server.multipleLiveServers</code>	true/false	false	This property defines whether the server publishes to multiple live servers. Note that this flag cannot be easily changed after the first start of the <i>Content Management Server</i> .
<code>cap.server.documentTypes</code>	URL/Path		This property defines where the server finds the XML file(s) containing the content type definitions. You can specify multiple files as a comma separated list or use Ant-style patterns with wildcards like <code>'*'</code> , <code>'?'</code> and <code>'**'</code> . Example: <code>config/contentserver/doctypes/**/*.*.xml</code> matches all XML files below the <code>config/contentserver/doctypes</code> directory.
<code>cap.server.resourcecache.size</code>	Int	60000	This property defines the resource cache size, that is, the number of resources the server holds in memory. This value should sometimes be adapted to the increasing number of resources in the actual

Property	Value	Default	Description
			working set. If the value is too small, the server does not perform well. One resource needs about 2kB of heap space.
cap.server.license	URL/Path	properties/correm/license.zip	This property defines where the server finds the license file.
cap.server.login.bouncers	URL/Path		This property points to the optional login bouncer configuration. A login bouncer can grant or deny access to the <i>Content Server</i> based on the characteristics of the user and the set of currently logged in users.
cap.server.useStrictWorkflow	true/false	false	This property enforces the strict workflow mode. That is, the approver of a resource must be different from the editor. This is checked independently of the workflow engine, and should only be used in cases where a custom workflow definition is not an option.
cap.server.checkUniqueDbAccess	true/false	false	This property determines whether to check for another server that is running concurrently on the same database on server startup. For compatibility, this property defaults to false, but it is recommended to set it to true in order to prevent data corruption.
cap.server.uniqueDbAccessWriteInterval	Int	0	If unique DB access is checked and if this property is positive, this property determines the number of seconds between two writes of the current timestamp to the database to indicate the liveness of the server; if 0 or negative the server neither writes a timestamp regularly nor expects a timestamp to be written; this value defaults to 0.

Property	Value	Default	Description
<code>cap.server.user.cache.size</code>	Int	500	This property defines the size of the user cache. It limits the maximum number of users which can be found in one search for users in the user window of the <i>Site Manager</i> . Set the property to the size of the largest user search you want to perform, or the number of concurrently working users, whichever is greater.
<code>cap.server.group.cache.size</code>	Int	500	This property defines the size of the group cache. It limits the maximum number of groups which can be found in one search for groups in the user window of the <i>Site Manager</i> . Set the property so that all groups connected to rights can be cached in memory.
<code>cap.server.rights.cache.size</code>	Int	3000	This property defines the size of the rights cache. This cache stores the results of right calculations per resource, content type and member. If you have lots of different resources, content types and users you might need to adapt the value of the property. Check the proper size of the cache by examining the cache misses and faults in the log. To activate the log output of the rights cache set the <code>cap.server.rights.cache.status.interval</code> property to a value larger than zero.
<code>cap.server.member.folderrightscache.size</code>	Int	1000	This property defines the size of the folder-specific rights cache. This cache stores the results of right calculations per folder and member, aggregating the results for all content types. This cache might help custom code using APIs other than the <i>Unified API</i> , but mainly it affects the performance of the <i>Site Manager</i> in rare cases. Change this setting only if

Property	Value	Default	Description
			you observe the method <code>getRights(MemberKey)</code> in thread dumps of a slow <i>Content Server</i> .
<code>cap.server.allowSyntheticReplay</code>	true/false	true	Whether it is allowed for clients to request a synthetic replay of the content repository, for example using the constant <code>Timestamp.SYNTHETIC_REPLAY</code> of the <i>Unified API</i> . This is a very expensive operation that is rarely used except when setting up a <i>Replication Live Server</i> from scratch.
<code>cap.server.rights.cache.status.interval</code>	Int	0	This property defines the interval (in seconds) at which log output of the rights cache is written. "0" means, that no log output is written.
<code>cap.server.blob.channel.timeout</code>	Int	60	This property sets the timeout for streaming blobs to and from the database in seconds. In general, you don't have to change the default value. It is provided for exceptional cases, when the connection to the database is unreliable. If you deploy the <i>Content Server</i> as a web application, you have to use the connection timeout setting of the servlet container. In XPath notation this is <code>/Server/Service/Connector@connectionTimeout</code> in <code>server.xml</code> .
<code>cap.server.init.runlevel</code>	String	online	The initial runlevel that the server will try to reach on startup. Possible runlevels are: online, administration, maintenance. This property does not override the default behavior of the <i>Replication Live Server</i> for the initial replication as described in Section 3.10.1, "Installing the First Replication Live Server" [54] . The property is effect-

Property	Value	Default	Description
			ive only in the case of web application deployment.
cap.server.maximumStartupDelay	int	60	The maximum time after which the Content Server is treated as initialized. Dependent web applications will be started when the Content Server has reached its initial runlevel or after this time. Note that large values may delay the startup of the container by that time.
cap.server.encryptpasswords.keyfile	String		The location of the key generated by <i>cm encryptpasswords</i> . If empty, defaults to <code>etc/keys/<database name>.<dbuser>.rijndael</code>
cap.server.initialPassword.<user>	String		The initial password to set for the default user with the indicated name, taken from the list in Section "Standard Groups and Users" [160] . This password is set when the server is started for the first time. You can change the passwords later on at any time.

Properties for SSL connection:

Property	Value	Default	Description
cap.server.https.keystore	Path		The path to the KeyStore file which contains the certificate for the connection. The entry <code>/Server/Service/Connector@keystoreFile</code> in <code>server.xml</code> (for a separate SSL connector) has to have the same value as this property.
cap.server.https.password	String		The password of the KeyStore. The entry <code>/Server/Service/Connector@keystorepass</code> in <code>server.xml</code> (for a separate SSL

Table 5.2. contentserver.properties

Property	Value	Default	Description
			connector)has to have the same value as this property.
cap.server.https.key password	String		The key to the certificate. The entry /Server/Service/Connector@keyPass in server.xml (for a separate SSL connector)has to have the same value as this property.

Properties to enable/disable the search service:

Property	Value	Default	Description
cap.server.search.enable	true or false	false	if true full text search is enabled.

Table 5.3. contentserver.properties

Use the home folder as it is defined above only for your personal files (queries, preferences etc.). Don't use it for content that will be published.



5.2 Configuration in publisher.properties

The publisher is a sub process running in the *Content Management Server*. When the publisher publishes resources, it creates new sessions to the *Content Management Server* and the *Master Live Server*. You configure the publisher in the file `publisher.properties` of the *Content Management Server* installation.

If you change user name, domain and password for the publisher, you have to make these changes in the user administration of the appropriate server too. To change the settings of the *Master Live Server*, you need to connect to the *Master Live Server* with the *Site Manager*. Then you can use the user administration (see [Section 3.16.6, “Managing Users” \[180\]](#)) for the changes.

When defining publisher priorities as described subsequently, the idea is to prioritize sessions with a higher portion of user interaction. Higher numerical values indicate a publication that has to be processed first. By convention, publication priorities should be nonnegative integers. Usually, the default priorities ensure a smooth publisher operation. When applying a change, two unmodifiable values have to be taken into account: Expedited server publications of rights rules run at a priority of 100. All customized priorities must be strictly less than that value. Accelerated *CoreMedia Editor* publications run at a priority of 80. Automated publications should normally receive a lower priority.

In the properties given below, some properties take a different form when using *CoreMedia Multi-Site Management*. In that case, an index number is used to indicate the publication target that is configured using this property value. The index numbers are consecutive integers starting with 1 and running up to the number of publication targets.

Property	Value	Default	Description
<code>publisher.local.user</code>	string	publisher	The user name for logging in locally to the <i>Content Management Server</i> .
<code>publisher.local.domain</code>	string		The domain for logging in locally to the <i>Content Management Server</i> .
<code>publisher.local.password</code>	string	publisher	The password for logging in locally to the <i>Content Management Server</i> .
<code>publisher.target.ior.url</code>	http:// LiveServer Master Computer	N/A	The URL where the publisher can obtain the IOR of the <i>Master Live Server</i> (required, single-site mode only).

Table 5.4. *publisher.properties*

Property	Value	Default	Description
	<code>media/ior</code>		
<code>publisher.target.user</code>	string	publisher	The user name for logging in to the <i>Master Live Server</i> (single-site mode only).
<code>publisher.target.domain</code>	string		The domain for logging in to the <i>Master Live Server</i> (single-site mode only).
<code>publisher.target.password</code>	string	publisher	The password for logging in to the <i>Master Live Server</i> (single-site mode only).
<code>publisher.target.<n>.{ior.url,user, domain,password}</code>	(see above)	(see above)	Login data for the publisher. These values correspond to the four properties above, but different numeric values of <code>n</code> are used to distinguish publication targets (multi-site mode only).
<code>publisher.target.<n>.name</code>	intranet	publication-target-<n>	The permanent and unique name of the publication target. Once set, it should never be changed, as this name is used for target identification in the APIs and in JMX (multi-site mode only).
<code>publisher.target.<n>.display.name</code>	Intranet	value of <code>publisher.target.<n>.name</code>	The display name is shown to users when no localized information about a publication target is available; display names, too, should be unique, but they may well change to better illustrate the current uses of a publication target (multi-site mode only).
<code>publisher.target.<n>.folders</code>	comma-separated list of names and/or ids		The base folders that are assigned to a publication target. This property typically references exactly

Property	Value	Default	Description
	of base folders, for example, in tranet, down load, 4711		one top-level folder, either by name or by its numerical id. If more than one site is generated from a single <i>Live Server</i> , multiple top-level folders may be given, separated by commas. When indicating a folder by name, that name is blocked for rename operations on the top-level folder. Once you have assigned a folder to a publication target, it must not be reassigned to another target. Doing so would result in inconsistencies between <i>Content Management Server</i> and <i>Master Live Server</i> (multi-site mode only).
publisher.enableBypassPreviews	true / false	true	Whether publication previews bypass ("true") or not ("false") the publication queue for faster response times.
publisher.autoextend.latestApprovedVersion	true/false	false	<p>This property affects the automatic extension of publication sets. The default behavior ("false") is as follows:</p> <ul style="list-style-type: none"> → If the referenced content item is not published yet, publish its earliest approved version. → If the referenced content item is already published, do nothing <p>If set to "true", not the earliest but the latest (= newest) approved version is published.</p>

Property	Value	Default	Description
<code>publisher.autoextend.updateLinkedDocuments</code>	true/false	false	This property affects the automatic extension of publication sets. By default, ("false") only direct linked and approved content items are added to the publication set. If set to "true", all linked content items are recursively added to the publication set. The recursion stops when a version is already published and may lead to surprisingly large publication sets. Setting <code>updateLinkedDocuments</code> implicitly also sets <code>latestApprovedVersion</code> .
<code>publisher.autoextend.maxFailures</code>	int	0	The maximum number of implicitly added content items and folders whose publication may fail before no further content items and folders may be added implicitly. If set to 0 (the default), the publisher never stops adding content items and folders.
<code>publisher.destroyIntermediateVersions</code>	off, strict, dumb	strict	Whether intermediate content item versions between two publications will be destroyed or not on the <i>Content Management Server</i> . Example for dumb mode: Version 1 of content item A was published. In the meantime, the versions 2, 3, 4 and 5 have been created. When you now publish version 5, the versions 2, 3, 4 are destroyed and only version 1 and 5 remain on the <i>Content Server</i> .

Property	Value	Default	Description
			<p>Old flags <code>true</code> and <code>false</code> are supported, too. <code>false</code> maps to <code>off</code> while <code>true</code> maps to <code>strict</code>.</p> <p><i>Available Modes</i></p> <p>off Turns destruction of intermediate versions off.</p> <p>dumb Simply destroys all versions between the currently published one and the previously published one; if you run into performance issues on publication or you do not use a multi-site set up this should be chosen. Mind that for multi-site set up this setting might break translation states.</p> <p>strict Destroys all versions between the currently published one and the previously published one, but only if the versions are not referenced by master/masterVer-</p>

Property	Value	Default	Description
			<p>sion properties. This ensures that for a multi-site set up your translation state will be valid. This setting is recommended for multi-site set ups.</p>
<code>publisher.destroyOlderVersionsOnLiveServers</code>	true / false	true	Whether older published versions should be destroyed ("true") on the <i>Master Live Server</i> . That is, only two published versions (the current and the one before) of each content item remain on the <i>Master Live Server</i> . Only change if you have a valid reason.
<code>publisher.priority.gui.editor</code>	0...99	60	The priority of publications requested through the <i>Site Manager</i> (default 60).
<code>publisher.priority.uapi</code>	0...99	40	The priority of publications requested through the Unified API (default 40).
<code>publisher.priority.jython</code>	0...99	20	The priority of publications requested through Jython code (obsolete, default 40).
<code>publisher.priority.importer</code>	0...99	20	The priority of publications requested through an importer (default 20).
<code>publisher.priority.unknown</code>	0...99	20	The priority of publications of unknown origin.

5.3 Configuration in `sql.properties`

In the file `sql.properties` the connection of the *Content Server* to the database via a JDBC driver is configured. The JDBC driver JAR files must be installed in the `lib-` directory of the *CoreMedia CMS* installation.

5.3.1 Overview of all Properties

The meaning of the properties of the `sql.properties` file are described in this section. Please read the other sections of this chapter for database specific configuration.

Connect to the Database

These properties are used to connect to the database.

	Value	Default	Description
<code>sql.store.driver=oracle.jdbc.driver.OracleDriver</code>	String		The JDBC driver used to connect to the database.
<code>sql.store.url=jdbc:oracle:thin@<HostName>:<Port>:CM</code>	String		The URL of the database to connect to. Replace <code><HostName></code> and <code><Port></code> with the appropriate values of the database host. Don't replace <code><HostName></code> with "localhost", this may cause problems with some JDBC drivers.
<code>sql.store.user=username</code>	String		The user name at the database.
<code>sql.store.password=password</code>	String		The password of the user at the database.
<code>sql.store.logDriverMessages=true</code>	true/false	false	Write all messages of the JDBC driver to the log file defined in <code>capserv.properties</code> .
<code>sql.store.driver.jdbc=</code>			Enter the version of the JDBC API supported by the JDBC driver. An empty property uses JDBC 1.0.

Table 5.5. Properties for configuration of database connection

	Value	Default	Description
<code>sql.store.isolation=</code>			Define the transaction isolation level. An empty value uses the driver default setting.

Changes of the Database Schema

These properties define how the *Content Servers* deal with content type changes. All checks and changes are performed at starting time of the *Content Servers*.

	Value	Default	Description
<code>sql.schema.check Columns=true</code>	true/false	true	Setting the property <code>sql.schema.checkColumns</code> to "true", causes the <i>Content Server</i> to compare the content type definition with the existing database schema for missing columns and matching widths of String properties. If there are differences, it depends on the setting of <code>sql.schema.alterTables</code> if the <i>Content Server</i> refuses to start ("false") or if it adds and changes the columns automatically ("true"). Checking the tables consumes a considerable amount of time, so that the server starts up more slowly. If <code>sql.schema.checkColumns</code> is set to "false", the <i>Content Server</i> will not check the columns. If there are differences, you will run into <i>Content Server</i> exceptions later on.

Table 5.6. Properties for configuration of database schema

	Value	Default	Description
<code>sql.schema.createTable=true</code>	true/false	true	The <i>Content Server</i> always checks if tables for the content types are missing. Setting the property to "true", causes the <i>Content Server</i> to create missing tables for new content types. If the setting is "false" and there are missing tables the <i>Content Server</i> refuses to start
<code>sql.schema.alterTable=false</code>	true/false	false	Setting the property to "true", causes the <i>Content Server</i> to automatically add database columns for newly defined content type properties and to fix wrong widths of String properties. This will only work if you set <code>sql.schema.checkColumns</code> to "true".
<code>sql.schema.createDropIndexes=false</code>	true/false	false	Setting the property to "true", causes the <i>Content Server</i> to automatically create and drop indexes on content type properties, according to the <code>Index</code> attribute in the <code>doctypes.xml</code> . This flag only affects existing columns. For newly added columns and tables, an index is always created if the <code>Index</code> attribute is set.

Configure the XML Property Conversion

Since *SC/ 4.1*, the new `coremedia-richtext-1.0.dtd` was introduced to replace the previously used `coremedia-sgmltext.dtd`. In addition, it is now possible to use own DTDs. In general, text which conforms to the `coremedia-sgmltext.dtd` will be automatically converted into `coremedia-richtext-1.0.dtd` conform text without any additional configuration. In some cases (custom XML

formats, for example), more work is necessary (please read also the [Operations Basics Manual]).

Table 5.7. Properties for the configuration of the XML property conversion

	Value	Default	Description
<code>sql.store.convert.<DocumentType>.<PropertyType>=<com.customer.XMLConverter></code>			Converters which are used to convert custom XML formats which have been defined for the obsolete <code>coremedia-sgmltext.dtd</code> .
<code>sql.store.convert.correctRichText=true</code>	true/false	true	The editor in versions before <i>SCI 4.1.38</i> created XML text, which was not valid according to the <code>coremedia-rich-text-1.0.dtd</code> . If you have created content with versions before <i>SCI 4.1.38</i> , set the property to "true" to correct these errors (CoreMedia recommends to always use "true").
<code>sql.store.replaceSubstitute=false</code>	true/false	false	There are problems with zOS DB2 databases to store characters that are not in the databases character set. Such characters are converted to the character with the code "0x001A" upon read. If the property is set to "true", this character will be replaced with the character defined in <code>sql.store.substituteCharacter</code> , thereby avoiding the replacement character that is illegal in XML texts.
<code>sql.store.substituteCharacter=<character></code>	Char	'?'	The character, with which "0x001A" should be replaced.

Configure the blob and SgmlText Collector

The Collector is a component which erases SgmlText and blobs from content versions which have been deleted.

	Value	Default	Description
<code>sql.store.collector.initialDelay=<n></code>	Int	600	Waits <n> seconds before starting the first collection.
<code>sql.store.collector.delay=<n></code>	Int	86400	The delay between two collections in seconds.
<code>sql.store.collector.startTime=<n></code>	Int	-1	The time of the day, in seconds from 0:00h in the default time zone, when the collector should run every day. If set to -1, it does not run at a fixed time. The initial delay is always enforced as a minimal delay after server startup.
<code>sql.store.collector.suspend=false</code>	Boolean	false	If set to <code>true</code> , unused blobs will not be deleted in the blob store. This should be used during backup (see Section 3.9.1, "Backup Strategy" [50]) if you have a non-transactional blob store.
<code>sql.store.collector.blob.preservationperiod=<n></code>	Int	86400	The time in seconds, a blob, that has no reference to a resource, will be preserved.

Table 5.8. Properties for configuration of blob and SgmlText collector

Configure the SQL Connection Pool

Property	Value	Default	Description
<code>sql.pool.logScheduleMessages=false</code>	true/false	false	If the property is "true", write messages of the connection pool to the log file defined in <code>content.server.properties</code> .

Table 5.9. Properties to configure the SQL connection Pool

Property	Value	Default	Description
<code>sql.pool.logVerbose=false</code>	true/false	false	If the property is "true", more debug messages will be generated.
<code>sql.pool.logQueries=false</code>	true/false	false	If the property is "true", messages concerning queries (search of content item versions) will be generated.
<code>sql.pool.logQueryStatements=false</code>	true/false	false	If the property is "true", SQL statements concerning queries will be written to the log.
<code>sql.pool.minConnections=<n></code>	Int	2	Minimum number <n> of connections to the database.
<code>sql.pool.maxConnections=<n></code>	Int	4	Maximum number <n> of connections to the database.
<code>sql.pool.maxQueries=<n></code>	Int	4	Maximum number <n> of connections used for queries, that is, the maximum number of parallel queries.
<code>sql.pool.reaperInterval=<n></code>	Int	120	The interval <n> in seconds in which it is checked if connections can be closed.
<code>sql.pool.reaperTimeout=<n></code>	Int	180	The time <n> in seconds a connection must be idle before it will be closed.
<code>sql.pool.validatorInterval=<n></code>	Int	300	The interval <n> in seconds in which existing connections will be checked for function.
<code>sql.pool.validatorTimeout=<n></code>	Int	120	The time <n> in seconds a connection must be idle before it will be checked for function.
<code>sql.pool.checkTimeout=<n></code>	Int	5	Maximum time <n> in seconds the check is allowed to take.

Property	Value	Default	Description
<code>sql.pool.openTimeout=<n></code>	Int	30	Maximum time <n> in seconds the opening of connections is allowed to take.
<code>sql.pool.closeTimeout=<n></code>	Int	40	Maximum time <n> in seconds the closing of connections is allowed to take.
<code>sql.store.preparedStatementCacheSize=<n></code>	Int		If set, denotes the maximum number of prepared statements that is kept open per database connection.

5.4 Configuration in replicator.properties

The property file `replicator.properties` configures the replicator process in a *Replication Live Server*.

Property	Value	Default	Description
<code>replicator.tmpDir</code>	Path	<code>var/tmp</code>	The path to the folder where the <i>Replication Live Server</i> should write temporary data during replication. The path is relative to the <code>WEB-INF</code> directory of the replicator web application, so you will probably use an absolute path. During initial replication of a complete repository, a huge amount of data will be written to this directory, in the order of the repository size. So, this directory should be located on a suitable file system with enough space left. During normal operation, only newly published blob content will be buffered on local disk
<code>replicator.publicationIORUrl</code>	URL in format <code>http://<MasterLiveServerComputerName>/oe-media/ior</code>		Sets the location where the <i>Replicator</i> can find the IOR of the <i>Master Live Server</i> .
<code>replicator.{user,domain,password}</code>	String		Configures the user name, domain and password to authenticate against the <i>Master Live Server</i> .
<code>replicator.chunkingThreshold</code>	int	0	Sets the maximum number of events that is fetched from the <i>Master Live Server</i> in one chunk during startup; 0 indicates

Table 5.10. *replicator.properties*

Property	Value	Default	Description
			no limit. Lowering the threshold will reduce main memory consumption at the cost of startup times.
<code>replicator.Packager.flushSize</code>	int	1000	Sets the maximum number of events that the packager holds in main memory before flushing to disk. Lowering the flush size will reduce main memory consumption at the cost of write performance.
<code>replicator.{IncomingCounter,Preparer,PrepareLogger,Packager,PackageLogger,Executor,ExecutionLogger,CompletedCounter}.channelSize</code>	int	1000	Sets the channel sizes for the event pipeline before the given stage; 0 indicates unlimited queues. In rare cases, this setting might lead to out of memory errors. Lowering the queue sizes will reduce main memory consumption at the cost of write performance.
<code>replicator.startupTimeout</code>	true/false	false	Defines if the <i>Replication Live Server</i> waits on start for the <i>Replicator</i> to connect to the <i>Master Live Server</i> (true).
<code>replicator.enable</code>	true/false	true	Defines if the <i>Replicator</i> should be started on start of the <i>Replication Live Server</i> (true).
<code>replicator.preventOnlineSwitch</code>	true/false	false	If set to "true" the <i>Replication Live Server</i> will replicate the content but will not be switched online afterwards.
<code>replicator.maxAcceptedLag</code>	int	100	The <i>Replication Live Server</i> is offline after a consistent replication but it is more than the given number of events behind

Property	Value	Default	Description
			the current timestamp, then the Replication Live Server will not go online until it catches up.
<code>replicator.restartReplicatorOnError</code>	true/false	true	Restart the <i>Replicator</i> if an error occurs on <i>Replication Live Server</i> side (true). Otherwise, the <i>Replicator</i> will be stopped.
<code>replicator.shutdownServerOnError</code>	true/false	false	Stop the <i>Replication Live Server</i> if an error occurs on <i>Replication Live Server</i> side.
<code>replicator.logEvents</code>	true/false	false	All repository events will be logged (true).
<code>replicator.autoRestart</code>	true/false	true	Restarts the <i>Replicator</i> sessions if lost (true).
<code>replicator.checkStream</code>	true/false	true	Defines if the <i>Replication Live Server</i> checks the event queue for connection (true). The event stream is not checked during the initial replication.
<code>replicator.checkTimeout</code>	int	300	The time in seconds after which the <i>Replication Live Server</i> checks the event queue for connection.

5.5 Configuration in capclient.properties

The file `capclient.properties` configures the IOR URL of the server as well as the global timezone.

Property	Value	Default	Description
<code>cap.client.server.ior.url</code>	URL format ht tp://<server>:<port> /core media/ior		This property determines where to get the IOR of the <i>contentserver</i> . <server> must be the name of the <i>Content Server</i> host. For <port> you have to set the server's web server HTTP port. Both values must match with the corresponding value in the <code>contentserver.properties</code> file on the server.
<code>cap.client.timezone.default</code>	TimeZone	Europe/Berlin	This parameter determines the used timezone. The standard value is <i>Europe/Berlin</i> . More time zones are documented in the class <code>java.util.TimeZone</code> .

Table 5.11. *capclient.properties*

5.6 Reference of webdav.properties

This section contains a description of the properties of the file `webdav.properties`. The file contains the configuration of the WebDAV Server such as authentication settings, exported content types and session timeout.

Authentication

Property	Value	Default	Description
<code>authentication.scheme.<n></code>	Basic or Digest		<p>This property configures the supported methods of HTTP/HTTPS authentication. This determines whether the password of the client will be submitted encrypted (<code>Digest</code>) or in plain text (<code>Basic</code>).</p> <p>The WebDAV clients of Windows Vista and Windows 7 support Basic authentication only for secure HTTPS connections. Digest authentication can be used with HTTP connections but does not support authentication of LDAP users. Therefore, it is recommended to use Basic authentication with HTTPS:</p> <pre>authentication.scheme.1=Basic</pre> <p>You can configure the WebDAV Server to support both Basic and Digest but note that most clients will always choose the first configured authentication method.</p> <pre>authentication.scheme.1=Basic</pre>

Table 5.12. HTTP authentication

Property	Value	Default	Description
			authentication.scheme.2=Digest

Exported Content Types and Properties

Property	Value	Default	Description
filesystem.documentFileType	file or folder	file	The default mapping used to export content items. If this is set to <code>folder</code> , all content types exported without a <code>filesystem.export.<n>.fileType</code> property are mapped to WebDAV folders. Otherwise, they are mapped to WebDAV files.
filesystem.appendFileExtension	true or false	true	If true, a file type extension is appended to all content items exported as files. The extension is derived from the MIME type of the exported property and must be configured in <code>mime.properties</code> . When creating a new content file, the extension will not show up in the content's name but will only be used to set the MIME type of the stored blob field. For content file creation, the MIME type and extension must be configured in <code>mime.properties</code> in a way that it is possible to reconstruct the same extension from the MIME type without ambiguities.

Table 5.13. Properties for exported content types

Property	Value	Default	Description
			If <i>false</i> , no file extension is appended and exported files have exactly the same name as the <i>CoreMedia</i> content items they represent.
<code>filesystem.export.<n>.docType</code>	content type		Defines the exported content type. All content types that inherit from the specified content type are also exported. <n> is a number different for each content type. Properties with the same number <n> belong together.
<code>filesystem.export.<n>.fileType</code>	file or folder	the value of <code>filesystem.documentFileType</code>	The mapping used to export content items of the type with number <n>. If this is set to <i>folder</i> , the content type is exported as WebDAV folder, which contains files for its exported fields. Otherwise, it is exported as WebDAV file with the content of a single exported field.
<code>filesystem.export.<n>.property[.<m >]</code>	content field		This property defines the exported fields (also named properties) of the exported content type. For the file mapping, only one field in a content type can be exported. For the folder mapping, replace [.<m >] with different numbers to specify multiple exported fields.
<code>filesystem.export.<n>.extension[.<m >]</code>	extension		If a client creates a new WebDAV file or folder, a respective content item or folder is created in the <i>CoreMedia Content Server</i> . The type of a created item depends on the extension of the created WebDAV

Property	Value	Default	Description
			<p>file or folder. Set the value of this property to the extension of the file such as <code>.doc</code> for a Word document. Replace <code><n></code> with the number for the respective content type. To map multiple extensions to a single content type (for instance <code>jpg</code> and <code>jpeg</code>), use multiple properties with different numbers for <code>[.<m>]</code>. You can use an asterisk (<code>*</code>) to configure a rule for all not explicitly configured extensions. Note, that it is not possible to mix characters and such an asterisk.</p>
<code>filesystem.export.<n>.emptyDocument</code>	Path or numeric ID of a content item		<p>If the exported field of a content item is empty, a zero-byte file appears in the WebDAV folder. Some WebDAV clients may have problems with an empty file. You can configure the WebDAV Server to read the content from a property of a substitute content item in the repository. Replace <code><n></code> with the number of the exported content type and <code><path id></code> with the path or numeric ID of the substitute content item. The user needs read rights on this item. All further changes like save or delete are performed not on the substitute content item, but on the originally selected item with the initially empty field.</p>

For example:

```
filesystem.export.1.docType=WordFile
filesystem.export.1.property=data
filesystem.export.1.extension.1=.doc
filesystem.export.1.extension.2=.rtf
filesystem.export.1.emptyDocument=/substitute/word
filesystem.export.2.docType=Article
filesystem.export.2.fileType=folder
filesystem.export.2.property.1=text
filesystem.export.2.property.2=image
filesystem.export.2.extension=-Article
```

The example exports the content types *WordFile* and *Article*. Content items of type *WordFile* are exported as files with the content of the field *data*. Content items of type *Article* are exported as folders with two contained files for their fields *text* and *image*. When a WebDAV client opens an empty *WordFile* item, the content of the field *data* of the item */substitute/word* is returned instead. The WebDAV Server creates content items of type *WordFile* for files with extensions *.doc* and *.rtf* and content items of type *Article* for WebDAV folders with the extension *-Article*.

Localize Field Names

With the folder mapping fields of content items are mapped to WebDAV files below a folder which represents the item. By default, the names of the files are the same as the names of the fields of the content item with an additional extension. The names can be localized in the same way as in the *Site Manager*. Create additional files for the different locales in the same directory as the file *webdav.properties* such as *webdav_de.properties* and *webdav_en.properties*. Note, that only field localization is used from locale specific configuration files. Localize the file names with properties of the following format: *<DocumentType>/<FieldName>Label*. Replace *<DocumentType>* with the document type and *<FieldName>* with the name of the field.

For example:

```
Article/textLabel = Description
Article/imageLabel = Picture
```

Session timeout

Property	Value	Default	Description
filesystem.sessionTimeout	seconds	600	The WebDAV Server opens a session to the CoreMedia Content Server for each user. After the specified time (in

Table 5.14. Properties for the session timeout

Property	Value	Default	Description
			seconds) of inactivity the session is closed automatically and opened again when the user continues his work. There are two reasons why it is necessary to set this value: If the timeout value is small, the session is closed more often and the server response time increases. If the timeout value is too big, a license is consumed and cannot be used by another user. The system checks all <code>timeout/4</code> seconds, but at least all 2 minutes and at most all 5 seconds if the session timed out.

Information File

Property	Value	Default	Description
<code>filesystem.infoFile.name</code>	file name	<code>info.html</code>	This property configures the name of the information file.
<code>filesystem.infoFile</code>	false, true	true	This property suppresses the creation of the information file by choosing "false".
<code>filesystem.rememberMessages</code>	int	20	This property configures the number of error messages shown in the information file.

Table 5.15. The properties for the information file

Encoding

Property	Value	Default	Description
uriEncoding	UTF-8 or ISO-8859-1	UTF-8	Neither HTTP nor WebDAV specify the encoding which has to be used to transform the URI character representation into a byte sequence before hexadecimal escaping can be done. Most of the current WebDAV applications use ISO-8859-1 (latin-1) encoding. However, some clients require UTF-8 encoded URIs. The behavior of the WebDAV server can be configured, using this property. The setting is valid for all applications using the WebDAV server.
filesystem.default.encoding	UTF-8		Some WebDAV clients do not set a character encoding when sending request. The default encoding of the web application server will then be used instead for storing files. If this is a problem you might switch the default encoding for such a case with this property.

Table 5.16. Properties for character encoding

Customization

Property	Value	Default	Description
filesystem.listener	class name		Registers a custom implementation of <code>com.coremedia.cap.webdav.filesystem.FileSystemListener</code> which is noti-

Table 5.17. Properties for further customization

Property	Value	Default	Description
			fied when a content property is written with WebDAV. See the API documentation for details.
filesystem.orb.jndi	JNDI Name		Sets a new ORB to use which gets looked up via JNDI. For IBM WebSphere you should set this to <code>java:comp/ORB</code> .

5.7 Managed Properties

In this section, you will find tables with all properties and actions manageable via JMX. The entries below the *JMImplementation* key display information on the JMX implementation which will not be described here.

The information contained in the *Statistics* section are not described, because this information can only be interpreted by trained CoreMedia consultants who are familiar with the inner workings of the CoreMedia components.



Content Server Attributes

Attribute	Type	Description
<i>AppDesc</i>	Read-only	Description of the CoreMedia system, contains version information and the component name.
<i>FutureRunLevel</i>	Read-only	The future run level which will be reached after the SwitchRunLevel actions has succeeded.
<i>HostInfo</i>	Read-only	The computer which hosts the Content Server.
<i>HttpHost</i>	Read-only	The host name of the internal HTTP server of the Content Server.
<i>HttpPort</i>	Read-only	The port of the internal HTTP Server of the Content Server.
<i>InstDir</i>	Read-only	Installation directory of the Content Server.
<i>Java-Classpath</i>	Read-only	Java class path.
<i>JavaInstDir</i>	Read-only	Java installation directory.
<i>JvmInfo</i>	Read-only	Java JVM information.
<i>JvmProcessInfo</i>	Read-only	Java process information, the number of threads, free memory, used memory, total memory.
<i>OsInfo</i>	Read-only	Information about the operating system of the host.
<i>ResourceCacheCapacity</i>	Read/Write	The maximum size of the resource cache (in resources). Changes value of the property <code>cap.server.resourcecache.size</code> in <code>contentserver.properties</code> .

Table 5.18. JMX manageable attributes of the Content Server

Attribute	Type	Description
<i>ResourceCacheEntries</i>	Read-only	Number of resources entered into the resource cache in the last ResourceCacheInterval.
<i>ResourceCacheEvicts</i>	Read-only	Number of cache evicts in ResourceCacheInterval.
<i>ResourceCacheHits</i>	Read-only	Number of cache hits in ResourceCacheInterval.
<i>ResourceCacheInterval</i>	Read/Write	Interval in seconds after which the computation of the cache statistics starts again.
<i>ResourceCacheSize</i>	Read-only	The current cache size (in resources)
<i>RunLevel</i>	Read-only	The current run level of the Content Server.
<i>StrictWorkflow</i>	Read/Write	Use strict workflow (editor of content item, is another person as the approver of the item)
<i>SwitchCountdown</i>	Read-only	The remaining time of the SwitchRunLevel grace period. "-1" indicates no active grace period.
<i>Uptime</i>	Read-only	Uptime of the server in [ms]
<i>RepositorySequenceNumber</i>	Read-only	The sequence number of the latest successful repository transaction, useful for comparing a Master Live Server's HighestSucceededSequenceNumber with a Replication Live Server's LatestIncomingSequenceNumber

Content Server Operations

Operation	Parameter	Description
<i>switchToRunLevel</i>	p1 - Runlevel to switch to p2 - Delay before switching in seconds	Change the run level of the Content Server. Possible run levels are: <ul style="list-style-type: none"> ➔ offline ➔ maintenance ➔ administration

Table 5.19. JMX manageable operations of the Content Server

Operation	Parameter	Description
		→ online
<code>abortRun- LevelSwitch</code>		Stop switching to another runlevel.

Publisher Attributes

Attribute	Type	Description
<code>AvPublPrevTime</code>	Read-only	Average publication preview time [msec] per publication preview.
<code>AvPublSize</code>	Read-only	Average publication size
<code>AvPublTime</code>	Read-only	Average publication time [msec] per publication.
<code>AvWaitTime</code>	Read-only	Average waiting time [msec] in the queue.
<code>DestroyIntVersions</code>	Read/Write	Deprecated flag for destroying intermediate content versions. Use <code>DestroyIntermediateVersions</code> instead. "true" maps to mode "STRICT" and "false" maps to the mode "OFF".
<code>DestroyIntermediateVersions</code>	Read/Write	Destroy intermediate content versions. Values are "OFF", "STRICT" and "DUMB"
<code>DestroyOlderVersionsOnLiveServers</code>	Read/Write	Keep only the newest version on the Live Server. Values are "true" and "false".
<code>EnableBypassPreviews</code>	Read/Write	Whether publication previews bypass ("true") or not ("false") the publication queue for faster response times.
<code>FailedPublCount</code>	Read-only	Number of failed publications.
<code>FailedPublPrevCount</code>	Read-only	Number of failed publication previews.
<code>LastPublDate</code>	Read-only	Date of the last publication in Unix format (milliseconds since 01.01.1970).
<code>LastPublDate</code>	Read-only	Date of the last publication in human readable format.
<code>LastPublResult</code>	Read-only	Result of the last publication (success or failed).
<code>LastPublSize</code>	Read-only	Size of the last publication (in resources).
<code>LastPublTime</code>	Read-only	Required time for last publication in [msec].

Table 5.20. JMX manageable attributes of the Publisher

Attribute	Type	Description
LastPublType	Read-only	Type of last publication (preview or publication)
LastPublUser	Read-only	User who started the last publication.
LastPublWaitTime	Read-only	Waiting time for last publication.
LocalDomain	Read-only	Domain for <i>Content Management Server</i> access.
LocalPassword	Read-only	User password for <i>Content Management Server</i> access.
LocalUser	Read/Write	User name for <i>Content Management Server</i> access.
PublCount	Read-only	Number of successful publications.
PublInterval	Read/Write	Interval (in seconds) after which the computation of the statistics starts again.
PublPrevCount	Read-only	Number of successful publication previews
PublicationTargetNames	Read-only	Names of the publication targets if Multi-site publishing is used.
QueueSize	Read-only	Queue size of the Publisher (in content items).

Multi-site Publication Targets

Attribute	Type	Description
DisplayName	Read/Write	The display name of the publication target.
Folders	Read/Write	The base folders which are assigned to the target. Names and IDs can be used. Numbers are always taken as IDs.
IorUrl	Read/Write	The URL of the <i>Master Live Server</i> where the <i>Publisher</i> gets the IOR for connecting.
Name	Read-only	The unique and permanent name of the publication target. Once set, it should never be changed.
Password	Read/Write	The password used by the <i>Publisher</i> to connect to the <i>Master Live Server</i> .

Table 5.21. JMX manageable attributes for Publication Targets

Publisher Operations

Operation	Parameter	Description
<code>setPriority</code>	p1 - Client p2 - priority	<p>Set the priority for the defined client. This operation sets the properties <code>publisher.priority.<client></code> of the <code>publisher.properties</code> file. The priority is an integer value from 0 to 100 and the client is an integer value from the following list:</p> <ul style="list-style-type: none"> ➔ 0 - The editor GUI ➔ 1 - The Unified API ➔ 2 - The Active Delivery Server ➔ 3 - Jpython-Scripts ➔ 4 - The Importer ➔ 5 - Utilities with Unified API ➔ 6 - Unknown clients
<code>getPriority</code>	p1 - Client	Get the priority for the defined client.

Table 5.22. JMX manageable operations of the Publisher

Replicator Attributes

Attribute	Type	Description
CompletedCount	Read-only	Total number of events that have been completed since server startup.
CompletedStartTime	Read-only	Date of the first completion of an event since server startup.
FirstCompletionDuration	Read-only	Difference in milliseconds between the first arrival and the first completion of an event since server startup.
IncomingCount	Read-only	Total number of events that have arrived since server startup.

Table 5.23. JMX manageable attributes of the Replicator

Attribute	Type	Description
LatestCompletedArrival	Read-only	Date of the latest completion of an event.
LatestCompletedSequenceNumber	Read-only	Sequence number of the latest completed event.
LatestCompletedStampedNumber	Read-only	Sequence number of the latest completion of a StampedEvent (indicates the end of a publication).
LatestCompletionDuration	Read-only	Difference in milliseconds between the latest arrival and the latest completion of an event since server startup.
LatestIncomingArrival	Read-only	Date of the latest arrival of an incoming event.
LatestIncomingSequenceNumber	Read-only	Sequence number of the latest incoming event.
LatestIncomingStampedNumber	Read-only	Sequence number of the latest incoming StampedEvent (indicates the end of a publication).
IncomingStartTime	Read-only	Date of the first arrival of an incoming event since server startup.
UncompletedCount	Read-only	The number of events to be processed (the difference between the number of incoming events and completed events).

Glossary

Blob	Binary Large Object or short blob, a property type for binary objects, such as graphics.
CAE Feeder	Content applications often require search functionality not only for single content items but for content beans. The <i>CAE Feeder</i> makes content beans searchable by sending their data to the <i>Search Engine</i> , which adds it to the index.
Content Application Engine (CAE)	<p>The <i>Content Application Engine (CAE)</i> is a framework for developing content applications with <i>CoreMedia CMS</i>.</p> <p>While it focuses on web applications, the core frameworks remain usable in other environments such as standalone clients, portal containers or web service implementations.</p> <p>The CAE uses the Spring Framework for application setup and web request processing.</p>
Content Bean	A content bean defines a business oriented access layer to the content, that is managed in <i>CoreMedia CMS</i> and third-party systems. Technically, a content bean is a Java object that encapsulates access to any content, either to <i>CoreMedia CMS</i> content items or to any other kind of third-party systems. Various <i>CoreMedia</i> components like the <i>CAE Feeder</i> or the data view cache are built on this layer. For these components the content beans act as a facade that hides the underlying technology.
Content Delivery Environment	<p>The <i>Content Delivery Environment</i> is the environment in which the content is delivered to the end-user.</p> <p>It may contain any of the following modules:</p> <ul style="list-style-type: none">→ <i>CoreMedia Master Live Server</i>→ <i>CoreMedia Replication Live Server</i>→ <i>CoreMedia Content Application Engine</i>→ <i>CoreMedia Search Engine</i>→ <i>Elastic Social</i>

	<ul style="list-style-type: none"> → <i>CoreMedia Adaptive Personalization</i>
Content Feeder	The <i>Content Feeder</i> is a separate web application that feeds content items of the CoreMedia repository into the <i>CoreMedia Search Engine</i> . Editors can use the <i>Search Engine</i> to make a full text search for these fed items.
Content item	In <i>CoreMedia CMS</i> , content is stored as self-defined content items. Content items are specified by their properties or fields. Typical content properties are, for example, title, author, image and text content.
Content Management Environment	The <i>Content Management Environment</i> is the environment for editors. The content is not visible to the end user. It may consist of the following modules: <ul style="list-style-type: none"> → <i>CoreMedia Content Management Server</i> → <i>CoreMedia Workflow Server</i> → <i>CoreMedia Importer</i> → <i>CoreMedia Site Manager</i> → <i>CoreMedia Studio</i> → <i>CoreMedia Search Engine</i> → <i>CoreMedia Adaptive Personalization</i> → <i>CoreMedia CMS for SAP Netweaver® Portal</i> → <i>CoreMedia Preview CAE</i>
Content Management Server	Server on which the content is edited. Edited content is published to the Master Live Server.
Content Repository	<i>CoreMedia CMS</i> manages content in the Content Repository. Using the Content Server or the UAPI you can access this content. Physically, the content is stored in a relational database.
Content Server	<i>Content Server</i> is the umbrella term for all servers that directly access the CoreMedia repository: <p><i>Content Servers</i> are web applications running in a servlet container.</p> <ul style="list-style-type: none"> → <i>Content Management Server</i> → <i>Master Live Server</i> → <i>Replication Live Server</i>

Content type	A content type describes the properties of a certain type of content. Such properties are for example title, text content, author, ...
Contributions	Contributions are tools or extensions that can be used to improve the work with <i>CoreMedia CMS</i> . They are written by CoreMedia developers - be it clients, partners or CoreMedia employees. CoreMedia contributions are hosted on Github at https://github.com/coremedia-contributions .
Controm Room	<i>Controm Room</i> is a <i>Studio</i> plugin, which enables users to manage projects, work with workflows, and collaborate by sharing content with other <i>Studio</i> users.
CORBA (Common Object Request Broker Architecture)	<p>The term <i>CORBA</i> refers to a language- and platform-independent distributed object standard which enables interoperation between heterogenous applications over a network. It was created and is currently controlled by the Object Management Group (OMG), a standards consortium for distributed object-oriented systems.</p> <p>CORBA programs communicate using the standard IIOP protocol.</p>
CoreMedia Studio	<p><i>CoreMedia Studio</i> is the working environment for business specialists. Its functionality covers all of the stages in a web-based editing process, from content creation and management to preview, test and publication.</p> <p>As a modern web application, <i>CoreMedia Studio</i> is based on the latest standards like Ajax and is therefore as easy to use as a normal desktop application.</p>
Dead Link	A link, whose target does not exist.
DTD	<p>A Document Type Definition is a formal context-free grammar for describing the structure of XML entities.</p> <p>The particular DTD of a given Entity can be deduced by looking at the document prolog:</p> <pre><!DOCTYPE coremedia SYSTEM "http://www.coremedia.com/dtd/coremedia.dtd"</pre> <p>There're two ways to indicate the DTD: Either by Public or by System Identifier. The System Identifier is just that: a URL to the DTD. The Public Identifier is an SGML Legacy Concept.</p>
Elastic Social	<i>CoreMedia Elastic Social</i> is a component of <i>CoreMedia CMS</i> that lets users engage with your website. It supports features like comments, rating, likings on your website. <i>Elastic Social</i> is integrated into <i>CoreMedia Studio</i> so editors can moderate user generated content from their common workplace. <i>Elastic Social</i> bases on NoSQL technology and offers nearly unlimited scalability.

EXML	EXML is an XML dialect supporting the declarative development of complex Ext JS components. EXML is Jangaroo's equivalent to Adobe Flex MXML and compiles down to Actions Script.
Folder	A folder is a resource in the CoreMedia system which can contain other resources. Conceptually, a folder corresponds to a directory in a file system.
Home Page	The main entry point for all visitors of a site. Technically it is often referred to as root document and also serves as provider of the default layout for all subpages.
IETF BCP 47	Document series of <i>Best current practice</i> (BCP) defined by the Internet Engineering Task Force (IETF). It includes the definition of IETF language tags, which are an abbreviated language code such as en for English, pt-BR for Brazilian Portuguese, or nan-Hant-TW for Min Nan Chinese as spoken in Taiwan using traditional Han characters.
Importer	Component of the CoreMedia system for importing external content of varying format.
IOR (Interoperable Object Reference)	A CORBA term, <i>Interoperable Object Reference</i> refers to the name with which a CORBA object can be referenced.
Jangaroo	<i>Jangaroo</i> is a JavaScript framework developed by CoreMedia that supports ActionScript as an input language which is compiled down to JavaScript. You will find detailed descriptions on the Jangaroo webpage http://www.jangaroo.net .
Java Management Extensions (JMX)	The Java Management Extensions is an API for managing and monitoring applications and services in a Java environment. It is a standard, developed through the Java Community Process as JSR-3. Parts of the specification are already integrated with Java 5. JMX provides a tiered architecture with the instrumentation level, the agent level and the manager level. On the instrumentation level, MBeans are used as managed resources.
JSP	JSP (Java Server Pages) is a template technology based on Java for generating dynamic HTML pages. It consists of HTML code fragments in which Java code can be embedded.
Locale	Locale is a combination of country and language. Thus, it refers to translation as well as to localization. Locales used in translation processes are typically represented as IETF BCP 47 language tags.
Master Live Server	The <i>Master Live Server</i> is the heart of the <i>Content Delivery Environment</i> . It receives the published content from the <i>Content Management Server</i> and makes it available to the CAE. If you are using the <i>CoreMedia Multi-Site Management Extension</i> you may use multiple <i>Master Live Server</i> in a CoreMedia system.

Master Site	A master site is a site other localized sites are derived from. A localized site might itself take the role of a master site for other derived sites.
MIME	With Multipurpose Internet Mail Extensions (MIME), the format of multi-part, multimedia emails and of web documents is standardised.
Personalisation	On personalised websites, individual users have the possibility of making settings and adjustments which are saved for later visits.
Projects	A project is a collection of content items in CoreMedia CMS created by a specific user. A project can be managed as a unit, published or put in a workflow, for example.
Property	<p>In relation to CoreMedia, properties have two different meanings:</p> <p>In CoreMedia, content items are described with properties (content fields). There are various types of properties, e.g. strings (such as for the author), Blobs (e.g. for images) and XML for the textual content. Which properties exist for a content items depends on the content type.</p> <p>In connection with the configuration of CoreMedia components, the system behavior of a component is determined by properties.</p>
Replication Live Server	The aim of the <i>Replication Live Server</i> is to distribute load on different servers and to improve the robustness of the <i>Content Delivery Environment</i> . The <i>Replication Live Server</i> is a complete Content Server installation. Its content is an replicated image of the content of a <i>Master Live Server</i> . The <i>Replication Live Server</i> updates its database due to change events from the <i>Master Live Server</i> . You can connect an arbitrary number of <i>Replication Live Servers</i> to the <i>Master Live Server</i> .
Resource	A folder or a content item in the CoreMedia system.
ResourceURI	A ResourceUri uniquely identifies a page which has been or will be created by the <i>Active Delivery Server</i> . The ResourceUri consists of five components: Resource ID, Template ID, Version number, Property names and a number of key/value pairs as additional parameters.
Responsive Design	Responsive design is an approach to design a website that provides an optimal viewing experience on different devices, such as PC, tablet, mobile phone.
Site	<p>A site is a cohesive collection of web pages in a single locale, sometimes referred to as localized site. In <i>CoreMedia CMS</i> a site especially consists of a site folder, a site indicator and a home page for a site.</p> <p>A typical site also has a master site it is derived from.</p>

Site Folder	All contents of a site are bundled in one dedicated folder. The most prominent document in a site folder is the site indicator, which describes details of a site.
Site Indicator	A site indicator is the central configuration object for a site. It is an instance of a special content type, most likely <code>CMsite</code> .
Site Manager	Swing component of CoreMedia for editing content items, managing users and workflows.
Site Manager Group	Members of a site manager group are typically responsible for one localized site. Responsible means that they take care of the contents of that site and that they accept translation tasks for that site.
Template	<p>In CoreMedia, JSPs used for displaying content are known as Templates.</p> <p>OR</p> <p>In <i>Blueprint</i> a template is a predeveloped content structure for pages. Defined by typically an administrative user a content editor can use this template to quickly create a complete new page including, for example, navigation, pre-defined layout and even predefined content.</p>
Translation Manager Role	Editors in the translation manager role are in charge of triggering translation workflows for sites.
User Changes web application	The <i>User Changes</i> web application is a <i>Content Repository</i> listener, which collects all content, modified by <i>Studio</i> users. This content can then be managed in the <i>Control Room</i> , as a part of projects and workflows.
Version history	A newly created content item receives the version number 1. New versions are created when the content item is checked in; these are numbered in chronological order.
Weak Links	<p>In general <i>CoreMedia CMS</i> always guarantees link consistency. But links can be declared with the <i>weak</i> attribute, so that they are not checked during publication or withdrawal.</p> <p>Caution! Weak links may cause dead links in the live environment.</p>
WebDAV	WebDAV stands for World Wide Web Distributed Authoring and Versioning Protocol. It is an extension of the Hypertext Transfer Protocol (HTTP), which offers a standardised method for the distributed work on different data via the internet. This adds the possibility to the CoreMedia system to easily access CoreMedia resources via external programs. A WebDAV enabled application like Microsoft Word is thus able to open Word documents stored in the CoreMedia system. For further information, see http://www.webdav.org .

Workflow	A workflow is the defined series of tasks within an organization to produce a final outcome. Sophisticated applications allow you to define different workflows for different types of jobs. So, for example, in a publishing setting, a document might be automatically routed from writer to editor to proofreader to production. At each stage in the workflow, one individual or group is responsible for a specific task. Once the task is complete, the workflow software ensures that the individuals responsible for the next task are notified and receive the data they need to execute their stage of the process.
Workflow Server	The <i>CoreMedia Workflow Server</i> is part of the Content Management Environment. It comes with predefined workflows for publication and global-search-and-replace but also executes freely definable workflows.
XLIFF	XLIFF is an XML-based format, standardized by OASIS for the exchange of localizable data. An XLIFF file contains not only the text to be translated but also metadata about the text. For example, the source and target language. <i>CoreMedia Studio</i> allows you to export content items in the XLIFF format and to import the files again after translation.

Index

A

- abstract content type, 213
- administrator group, 177
- administrator groups, 177

B

- BlobProperty, 202
- blobProperty, 204

C

- capclient.properties, 250
- cloudscape: no support, 240
- cm
 - validate-multisite, **104**
- CollectablePredicate, 121
- configuration, 54, 56
- Content Management Server Group, 178
- content server
 - JMX, 259
- content server groups, 178
- content types
 - abstract type, 213
 - adding, 218
 - adding properties, 221
 - deleting, 220
 - grammar, 211
 - inheriting, 213
 - name limitations, 210
 - names, 210
 - properties, 202
 - renaming, 219
 - renaming properties, 221
 - schema migration, 216
- contentserver.properties, 227
- CoreMedia CMS, 25

D

- database, 34, 36
- dateProperty, 203
- deleting content types, 220
- document types
 - creation, 208
 - deleting properties, 223
 - extending, 214
- DocumentTypes, 200

G

- garbage collection, 118
- Grammar, 209
- grammar, 211
- groups
 - administrator groups, 177
 - creation, 181
 - predefined, 160
 - workflow groups, 162

I

- indexing, 205
- intProperty, 202

J

- JMX, 259
- JMX management, 159

L

- LDAP integration, 77
- ldaps, 89
- license, 227
- link
 - weak, 205
- linkListProperty, 204
- Live Server Group, 179
- live server groups, 178

M

- managed properties, 259
- multi-site management, 23
- migrate to, 67

P

properties, 14, 247, 250
blobProperty, 204
dateProperty, 203
indexing, 205
intProperty, 202
linkListProperty, 204
maximum name length, 202
stringProperty, 203
types, 202
UTF-8 encoding, 206
xmlProperty, 203
publisher.properties, 234

R

Replication Live Server, 15, 17-18, 54-56, 60
rights
 computation of, 173
rule selection box, 186
runlevel, 48

S

schema migration, 216
Search Engine, 1
server utilities, 90, 134, 138
 cm groovysh, 129
 CM IOR, 97
 Groovy Shell, 129
sql.properties, 240
stringProperty, 203

T

troubleshooting, 19-21, 193, 198

U

users, 160
 assigning licenses, 187
 creation, 183
 management, 180
 predefined, 160
 rights management, 163

V

validate-multisite, **104**

W

weak link, 205
web application, 30
WebDAV, 194
 configuration, 194
 export as files, 191
 export as folders, 192
 info file, 193
 troubleshooting, 195
webdav.properties, 191, 251
workflow rule groups, 162

X

xmlProperty, 203