

COREMEDIA CONTENT CLOUD

Content Server Manual



Copyright CoreMedia GmbH © 2021

CoreMedia GmbH

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 GmbH.

Germany

Alle Rechte vorbehalten. CoreMedia und weitere im Text erwähnte CoreMedia Produkte sowie die entsprechenden Logos sind Marken oder eingetragene Marken der CoreMedia GmbH 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 GmbH reproduziert oder vervielfältigt werden. Unberührt hiervon bleiben die gesetzlich erlaubten Nutzungsarten nach dem UrhG.

Licenses and Trademarks

All trademarks acknowledged.
September 19, 2021 (Release 2104)

1. Preface	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	6
1.3.3. Documentation	6
1.3.4. CoreMedia Training	10
1.3.5. CoreMedia Support	10
1.4. Changelog	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 Environment	18
2.3. Multi-Master Publishing	23
2.4. Server Run Levels	25
2.5. Changelog	27
2.6. Editing Dates	28
3. Configuration and Operation	30
3.1. Deploying the Content Server	31
3.2. Configuring the Database	32
3.2.1. Specifying Tablespaces for the Content Server	33
3.2.2. Oracle Database	35
3.2.3. DB2	37
3.2.4. Microsoft SQL Server	39
3.2.5. PostgreSQL Database	40
3.2.6. MySQL Database	41
3.2.7. Replication at the Database Level	42
3.3. Configuring Blob Storage	43
3.4. Exclusive Locks	50
3.5. Configuring CORBA	51
3.6. Extending the Content Server	52
3.7. Starting the Server	53
3.8. Recovery of Content Server Databases	55
3.8.1. Backup Strategy	55
3.8.2. Recovery of a Content Management Server Data- base	56
3.8.3. Recovery of a Master Live Server Database	56
3.9. Administrating Replication Live Servers	60
3.9.1. Installing the First Replication Live Server	60
3.9.2. Installing further Replication Live Servers	61
3.9.3. Replication Live Servers Backups	62
3.9.4. Restoring from Replication Live Server Backup	62
3.9.5. Restoring from Master Live Server Backup	63
3.9.6. Removing a Replication Live Server	67
3.9.7. Analyzing the Replicator State	67

3.10. Administrating Multi-Master Publishing	72
3.10.1. Enabling Multi-Master Publishing	72
3.10.2. Configuring Multi-Master Publishing	72
3.10.3. Adding Publication Targets	74
3.10.4. Migrate to Multi-Master Management Extension	74
3.11. Truncate the ChangeLog	83
3.12. LDAP Integration	85
3.12.1. User Authentication	85
3.12.2. Configuration of UserProviders	93
3.12.3. LdapUserProvider	94
3.12.4. ActiveDirectoryUserProvider	96
3.12.5. Connecting LDAP Over SSL	98
3.12.6. Common Customizations	98
3.12.7. LDAP User and User Changes Application	99
3.13. Server Utility Programs	101
3.13.1. Information	102
3.13.2. Operation	134
3.13.3. Repository	191
3.14. JMX Management	210
3.15. User Administration	211
3.15.1. Predefined Users and Groups	211
3.15.2. User Rights Management	217
3.15.3. Administrator Groups	232
3.15.4. Content Server Groups and Users	233
3.15.5. Live Server Groups and Users	234
3.15.6. Assigning Licenses to Users	235
3.16. Troubleshooting	239
4. Developing a Content Type Model	241
4.1. Properties	242
4.2. Creating Content Type Definitions	250
4.2.1. Structure of Content Type Definitions	251
4.2.2. Inheriting Content Types	256
4.2.3. Attaching Properties to Existing Content Types	257
4.3. Schema Update	259
4.3.1. The Database Schema	259
4.3.2. Struct Properties	262
4.3.3. Adding Content Types	262
4.3.4. Renaming Content Types	263
4.3.5. Deleting Content Types	265
4.3.6. Adding Properties	267
4.3.7. Renaming Properties	268
4.3.8. Changing Properties	270
4.3.9. Deleting Properties	274
5. Reference	277
5.1. Configuration Property Reference	278
5.2. Managed Properties	280
Glossary	287
Index	293

List of Figures

- 2.1. Replication Live Server state diagram 16
- 3.1. Result of a command in Groovy Shell 169
- 3.2. CM SQL user interface 187
- 4.1. Schema of content type definitions 250
- 4.2. Extending content types 257

List of Tables

1.1. Typographic conventions	3
1.2. Pictographs	4
1.3. CoreMedia manuals	7
1.4. Changes	12
2.1. Master Live Server failure	19
2.2. Master Live Server deadlock	19
2.3. Database failure	20
2.4. Replication Live Server failure	21
2.5. Database failure	21
2.6. CAE failure	22
2.7. CAE deadlock	22
2.8. Different state dates	28
3.1. Attributes of the bean element	47
3.2. Condition classes which could be used in the <bean> element.	47
3.3. Attributes of the property element	48
3.4. Properties used to configure Multi-Master Management	72
3.5. Options of CapLoginModule	87
3.6. Options of the LdapLoginModule	87
3.7. CoreMedia services	88
3.8. NameLoginPredicate options	90
3.9. AttributeLoginPredicate options	91
3.10. JndiNameLoginPredicate options	92
3.11. Common options of server utilities	101
3.12. Options of dump	103
3.13. Options of license	106
3.14. The parameters of processorusage	108
3.15. Options of the publications utility	109
3.16. Options of the rules utility	111
3.17. Options of the session utility	111
3.18. Parameters of validate-link-type	112
3.19. Parameters of validate-multisite	114
3.20. Issues of validate-multisite	115
3.21. Options of the changepassword tool	137
3.22. Options of the clean recycle bin	139
3.23. Switches of the version collector	141
3.24. Switches of the version collector	143
3.25. Reference times for content-uuid-import for a million contents	151
3.26. Parameters of content-uuid-export	152
3.27. Parameters of generate-content-uuid-map	154
3.28. Parameters of content-uuid-import	156
3.29. Options of dbindex	158
3.30. Parameters of dumpusers	159
3.31. Variables for Groovy Shell	168
3.32. Options of jmxdump	170
3.33. Options of the events utility	174
3.34. Parameters of restoreusers	176
3.35. Options of runlevel	178

3.36. Schemaaccess actions	180
3.37. Parameters of the serverimport utility	182
3.38. Parameters of the serverexport utility	184
3.39. Options of tracesession	188
3.40. Parameters of the usedlicenses utility	190
3.41. Options of approve	192
3.42. Options of the bulkpublish tool	194
3.43. Options of destroy	195
3.44. Options of publish	197
3.45. Parameters of the publishall utility	198
3.46. Parameters of the republish utility	201
3.47. Parameters of the query utility	202
3.48. Identifiers and literals	205
3.49. Parameters of the queryapprove utility	207
3.50. Parameters of the querypublish utility	208
3.51. Parameters of the search utility	208
3.52. Standard groups	211
3.53. Users and their groups	212
3.54. Users and their groups in Content Management Server only	212
3.55. Default mapping of user rights	213
3.56. Workflow role groups	215
3.57. System users and usage	216
3.58. User rights	218
3.59. Example rule table	219
3.60. Rule to read a content item	220
3.61. Rule to create a content item	221
3.62. Rule for content item operations	221
3.63. Rules to move a content item	221
3.64. Rule to mark or (un)mark a content item for deletion	222
3.65. Rules to delete a content item	222
3.66. Rule to (dis)approve a content item	223
3.67. Rule to publish a content item	223
3.68. Rule to check in content items of other users	223
3.69. Rule to read folder properties	224
3.70. Rule to place approve or disapprove a folder	224
3.71. Rule to publish a folder	225
3.72. Rule to create subfolders	225
3.73. Rule to operate on subfolders	225
3.74. Rules to move a folder	226
3.75. Rule to supervise a content item	226
3.76. Rule to supervise content items in a folder	227
3.77. Rule to supervise a folder	227
3.78. Example rules for rights computation	228
3.79. Example for conflicting rules	228
3.80. Example rules to compute effective rights	230
3.81. Example rules with implicit navigate through right	230
3.82. Example rules with resolved navigate through right	231
3.83. Example rules with implicit READ right	231
3.84. Example rules with explicit READ right	231

- 3.85. Example rules with READ right withdrawn 232
- 4.1. Common Attributes 243
- 4.2. Attributes of a StringProperty field 243
- 4.3. Attributes of an XmlProperty field 244
- 4.4. Attributes of LinkListProperty 245
- 4.5. Attributes of XmlGrammar element 253
- 4.6. Attributes of the XMLSchema element 254
- 4.7. Attributes of the ImportGrammar element 255
- 4.8. System attributes of the content type table 260
- 4.9. How to delete properties of different type 275
- 5.1. JMX manageable attributes of the Content Server 280
- 5.2. JMX manageable operations of the Content Server 282
- 5.3. JMX manageable attributes of the Publisher 282
- 5.4. JMX manageable attributes for Publication Targets 284
- 5.5. JMX manageable operations of the Publisher 284
- 5.6. JMX manageable attributes of the Replicator 285

List of Examples

3.1. Oracle: create DB User	36
3.2. Add privileges for Oracle 12c user	36
3.3. Oracle: delete DB User	37
3.4. SQL properties for IBM DB2	38
3.5. Example for Bash Script for IBM DB2 Database Setup	38
3.6. Example configuration	45
3.7. Custom store configuration	46
3.8. The jaas.conf file	86
3.9. JAAS syntax	87
3.10. jaas.conf example	93
3.11. Example for Active Directory Server	93
3.12. Configuration of a second UserProvider	94
3.13. Configuration of java.naming properties	94
3.14. Base Distinguished Names	95
3.15. Execution of CM IOR with IOR of the CM Server	106
3.16. Result of repositorystatistics	109
3.17. Usage of validate-multisite	114
3.18.	136
3.19. Example of simple custom selector	146
3.20. Example for custom selector with parameter	146
3.21. Typical Usage Example	148
3.22. Usage of content-uuid-export	151
3.23. Usage of generate-content-uuid-map	154
3.24. Usage of content-uuid-import	155
3.25. Snippet of dumpusers output	158
3.26. Usage of dumpusers	159
3.27. Start JShell (Default)	165
3.28. Virtual Built-In Startup Script: COREMEDIA	166
3.29. Restrict to DEFAULT	166
3.30. Connection at Startup (System Properties)	166
3.31. Connection at Startup (Environment Variables)	166
3.32. Connection within JShell	166
3.33. Exit Code Handling	167
3.34. Save Startup Script	167
3.35. Usage of the events utility	173
3.36. Output of events	174
3.37. Snippet of dumpusers output	175
3.38. Usage of dumpusers	176
3.39. Usage of schemaaccess	179
3.40. CM sql command line operation	187
3.41. Usage of bulkpublish	193
3.42. Usage of publishall	198
3.43.	200
3.44. Query usage	201
3.45. Result of a query	203
3.46. Transformation of a legacy query	204
3.47. EBNF definition of the query language	204

- 3.48. Usage of search utility 208
- 3.49. Groups in Workflow Server properties 215
- 3.50. A sample login bouncer configuration 237
- 4.1. The base-doctypes.xml file 251
- 4.2. The specific-doctypes.xml file 251
- 4.3. Example of a content type definition 251
- 4.4. Using XML Schemas 254
- 4.5. Example for content type inheritance 256
- 4.6. Struct XML with link list property 262
- 4.7. Error message because of invalid link type 262
- 4.8. Log entries when string property has been changed 272
- 4.9. Log entries when an existing string property is newly observed 272
- 4.10. Log entries when an existing string property is not observed anymore
..... 273

1. Preface

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 \[30\]](#) describes how you configure and operate the *Content Servers*.
- [Chapter 4, Developing a Content Type Model \[241\]](#) describes how you develop a document type model for the *Content Servers*.
- [Chapter 5, Reference \[277\]](#) lists the configuration files of the *Content Servers*.

NOTE

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	cm systeminfo start
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	\	cm systeminfo \ -u user

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.
	Danger: The violation of these rules causes severe damage.

Table 1.2. Pictographs

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.

NOTE

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" \[6\]](#) 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" \[10\]](#) describes CoreMedia training. This includes the training calendar, the curriculum and certification information.
- [Section 1.3.5, "CoreMedia Support" \[10\]](#) 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" \[10\]](#)) 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:

<https://releases.coremedia.com/cmcc-10>

Refer to our [Blueprint Github mirror repository](#) for recommendations to upgrade the workspace either via Git or patch files.

NOTE

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 Section 3.1, "Prerequisites" in *Blueprint Developer Manual*.

License files

You need license files to run the CoreMedia system. Contact the support (see [Section 1.3.5, "CoreMedia Support" \[10\]](#)) 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:

<https://documentation.coremedia.com/cmcc-10>

The manuals have the following content and use cases:

Manual	Audience	Content
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.
Blueprint Developer Manual	Developers, architects, administrators	This manual gives an overview over the structure and features of <i>CoreMedia Content Cloud</i>. It describes 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. 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.
Connector Manuals	Developers, administrators	This manual gives an overview over the use cases of the eCommerce integration. It describes the deployment of the Commerce Connector and how to connect it with the CoreMedia and eCommerce system.
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
Deployment Manual	Developers, architects, administrators	This manual describes the concepts and usage of the CoreMedia deployment artifacts. That is the deployment archive and the Docker setup. You will also find an overview of the properties required to configure the deployed system.
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.
Frontend Developer Manual	Frontend Developers	This manual describes the concepts and usage of the Frontend Workspace. You will learn about the structure of this workspace, the CoreMedia themes and bricks concept, the CoreMedia Freemarker facade API, how to develop your own themes and how to upload your themes to the CoreMedia system.
Headless Server Developer Manual	Frontend Developers, administrators	This manual describes the concepts and usage of the <i>Headless Server</i> . You will learn how to deploy the Headless Server and how to use its endpoints for your sites.
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.
Multi-Site Manual	Developers, Multi-Site Administrators, Editors	This manual describes different options to design your site hierarchy with several languages. It also gives guidance to avoid common pitfalls during your work with the multi-site feature.
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.
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> .

Manual	Audience	Content
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>. The Site Manager is deprecated for editorial work.
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.
Studio User Manual	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>.
Studio Benutzerhandbuch	Editors	The Studio User Manual but in German.
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.
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.
Utilized Open Source Software & 3rd Party Licenses	Developers, architects, administrators	This manual lists the third-party software used by CoreMedia and lists, when required, the licence texts.
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.

Table 1.3. CoreMedia manuals

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 live online, 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:

Support checklist

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

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 `logback.xml` file.

Log files

Which Log File?

In most cases at least two CoreMedia components are involved in errors: the *Content Server* log files together with the log file from the client. If you know exactly what the problem is, 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 Section 4.7, "Logging" in *Operations Basics* for details.

1.4 Changelog

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

Section	Version	Description

Table 1.4. 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-Master 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

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 (see Section 4.2.5, “Properties for Replicator Configuration” in *Deployment Manual*).
- 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.

NOTE

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

NOTE

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*:

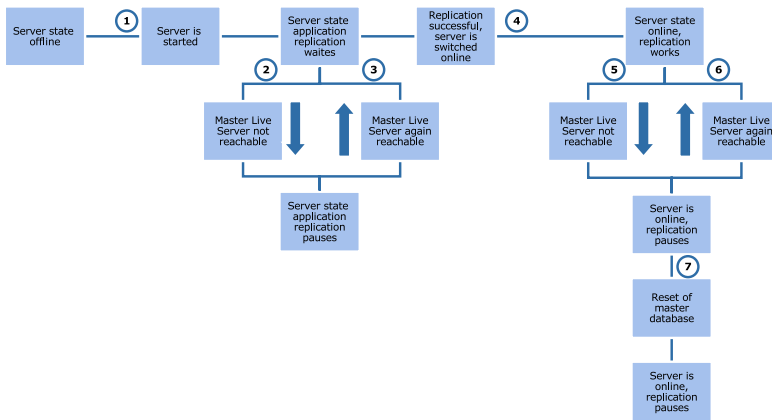


Figure 2.1. Replication Live Server state diagram

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. Afterwards, 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
- Edition date of content: This date field contains the date of the last created version. It is copied unchanged from the *Master Live Server*, indicating the time of the most recent publication of the last version. See [Section 2.6, "Editing Dates" \[28\]](#) for more details.

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	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. 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>.
Error correction	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.

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.
----------------	--

Master Live Server Deadlock

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. Replication 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 on 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 <i>CAE</i> 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 <i>CAE</i> detects the error and restarts the <i>CAE</i> (see the watchdog log).

Table 2.7. *CAE deadlock*

2.3 Multi-Master Publishing

Overview

If you are using *Multi-Master 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-Master Management* in different contexts.

Use Cases

Multiple Web Sites

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` 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.

2.6 Editing Dates

For content, versions and folders, different dates are stored in the database. You can access the dates via the CoreMedia APIs, for example with `Content.getCreationDate()` with the Unified API.

The different dates have sometimes different meanings on the three Content Server instances. [Table 2.8, "Different state dates" \[28\]](#) lists the different dates and describes the meaning of each date on each Content Server type.

Date	Content Management Server (CMS)	Master Live Server (MLS)	Replication Live Server (RLS)
creationDate	The date when the content or folder has been created.	The date when the content or folder has been published to the MLS for the first time.	The date when the content or folder has been replicated to the RLS for the first time. If the RLS is set up by initial replication, it may differ significantly from the date on the MLS.
modificationDate	The modificationDate is different for each individual server. It is the date of the last change of a content item, that is, for example, version updates, renames, moves.		
editionDate	The date of the creation of the last version on the CMS. A folder has no editionDate.	The date of the creation of the last version on the MLS. Since renames or moves do not generate new versions, the editionDate can become significantly different from the publicationDate, when renames or moves have been published.	The editionDate is replicated from the MLS to all RLS. Therefore, the editionDate is the same on MLS and all RLS.

Date	Content Management Server (CMS)	Master Live Server (MLS)	Replication Live Server (RLS)
publicationDate	The publicationDate is changed, whenever a version, a move or a rename is published. Since renames or moves do not generate new versions, the editionDate on the MLS can become significantly different from the publicationDate, when renames or moves have been published.	No publicationDate	No publication date

Table 2.8. Different state dates

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, Reference \[277\]](#). There you will find a description of all relevant properties, that you might use for configuration of the *CoreMedia CMS*.

3.1 Deploying the Content Server

The Content Server is deployed as a Spring Boot application.

3.2 Configuring the Database

NOTE

Please read the Supported Operating Environments document on the *CoreMedia Content Cloud* documentation <https://documentation.coremedia.com/cmcc-10> page in order to get the actually supported databases.



The general configuration of the database is described in Section 4.2.4, "Properties for the Connection to the Database" in *Deployment Manual*. Vendor-specific configuration can be done in the database dependent `<DBName>.properties` files.

NOTE

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 3.1, "Oracle: create DB User" \[36\]](#) and [Example 3.3, "Oracle: delete DB User" \[37\]](#))

Database Cluster

The Content Servers can be attached to a database cluster. While CoreMedia does not certify a specific cluster solution, clusters can be used as long as they use synchronous replication. That is, all transactions must be replicated to the other nodes before the end of the current transaction, so that all changes are visible even in case of a node failure. Otherwise, the memory state of the Content Server might differ from the database state after a failover.

For example, this might lead to a database error, with "parent key not found" in the logs, if a blob is referenced from the Content Server memory but does not exist in the database.



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.

NOTE

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.



3.2.1 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-table-options` 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.

CAUTION

The property `db.blob-table-options` 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-table-options=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-table-options=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-table-options=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 `doctypes.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>
```

CAUTION

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



3.2.2 Oracle Database

For the Oracle database you need the Oracle JDBC Thin Driver `oracle.jdbc.driver.OracleDriver`. Copy the driver file to the `lib` directory of the *Content Server*. 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.

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, 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<Instancename>.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` 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 3.1. 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 3.2. Add privileges for Oracle 12c user

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

Example 3.3. 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 is also configurable). 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$session s, v$sqltext 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

3.2.3 DB2

CoreMedia Content Servers can use IBM DB2 databases for their repositories. This section describes DB2 specific settings. For general advice see [Section 3.2, "Configuring the Database"](#) [32].

General Notes

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

JDBC Configuration

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

Configure the SQL properties as follows:

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

Example 3.4. SQL properties for IBM DB2

Database Configuration

The recommended configuration for the DB2 database is as follows:

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

Suggested values can be found in the example script below.

Sample Script for Database Setup

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

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

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

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

db2 update db cfg for ${DBNAME} using applheapsz 16384
db2 update db cfg for ${DBNAME} using stmtheap 60000
db2 update db cfg for ${DBNAME} using logfilsiz 131072
```

```

db2 update db cfg for ${DBNAME} using maxlocks 100
db2 update db cfg for ${DBNAME} using locklist 65536

db2set DB2_RR_TO_RS=yes

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

db2 grant CONNECT,DBADM,BINDADD,CREATE NOT FENCED_ROUTINE,\
  IMPLICIT_SCHEMA,LOAD,CREATE_EXTERNAL_ROUTINE,\
  QUIESCE_CONNECT,CREATETAB on database to user ${DBOWNER_USER}

db2 disconnect current
db2 detach

```

Example 3.5. Example for Bash Script for IBM DB2 Database Setup

3.2.4 Microsoft SQL Server

For the SQL Server database you must use the original JDBC driver supplied by Microsoft. Copy the driver file into the `lib` directory of the *Content Server* and configure the following settings:

```

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.

CAUTION

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.



3.2.5 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:

```
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 `<databasename>` with the name of the PostgreSQL database you created using the `createdb` command.

For PostgreSQL on Azure, the login user name is different from the database user name, and must be configured with an additional property:

```
sql.store.login-user-name=<DB-UserName>@<Domain>
```

CAUTION

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.

NOTE

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.

3.2.6 MySQL Database

The MySQL Connector/J provides JDBC support for MySQL. Please use an up to date JDBC4 driver for your version of MySQL. The driver can be found here: <https://dev.mysql.com/downloads/connector/j/>

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

```
sql.store.driver=com.mysql.cj.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 SCHEMA <user> CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
CREATE USER '<user>'@'localhost' IDENTIFIED BY '<password>';
CREATE USER '<user>'@'%' IDENTIFIED BY '<password>';
GRANT ALL PRIVILEGES ON <user>.* TO '<user>'@'%', '<user>'@'localhost';
```

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.

3.2.7 Replication at the Database Level

Databases provide various replication approaches, simplifying and speeding up the recovery after a database failure and in some cases allowing an automatic failover between database nodes.

For automatic failover, replication has to happen transparently, that is, the cluster of replication nodes has to behave exactly as a single database node. Mainly this requires that every change is persisted on either all database nodes or on a sufficient quorum before a transaction commits, ensuring that the change remains visible after a failover (synchronous replication). Consult the documentation of your database product to determine how it can be configured accordingly.

Even if your database product does not support synchronous replication, database replication can still be helpful as a way to automate and greatly speed up the backup/restore process. In such a case, the JDBC driver should not be configured for automatic failover, because the entries in the caches of the *Content Server* and its clients could become outdated. Instead, the *Content Server* and all of its clients should be restarted after a node in the database cluster does down. Solr instances should be refeeded. *Replication Live Servers* should be restarted if the *Master Live Servers* was affected.

3.3 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 3.2.1, "Specifying Tablespace for the Content Server" \[33\]](#)
- 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*.

If you add more media stores to the *Content Servers*, make sure that the media stores of the *Master Live Server* and all *Replication Live Server* are identical. Otherwise, the replicator would not be able to place the blobs into the same media store from which they were read. This would lead to inconsistencies, so that the server refuses to replicate such content.

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` before starting the backup process.

CAUTION

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

NOTE

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.



Blob storage is configured in the Spring application context of the *Content Server*. The predefined Spring bean `blobstore` of type `MediaStoreConfiguration` provides two customizable properties:

- `mediaStoreSelectors`: the list of media store selectors
- `mediaStores`: a map of media stores by their names

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 media store should be used. [Example 3.6, "Example configuration" \[45\]](#) 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.

```
<customize:prepend id="blobstoreCustomizer"
  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>
```

Example 3.6. Example configuration

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.jdbc.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.

You can check the effective order of custom media store selectors by looking at the INFO log output when the Content Server starts. Just search for a message containing "Configured media store selectors".

Custom media stores can be appended to the map in the `mediaStores` property of the `blobstore` bean.

The following example adds a custom media store named `videostore` and a media store selector to select that store for certain MIME types:

```
<customize:append id="addCustomBlobstore"
  bean="blobstore" property="mediaStores">
  <map>
    <entry key="videostore" value-ref="customVideoStore"/>
  </map>
</customize:append>

<customize:prepend id="addVideoStoreSelector"
  bean="blobstore"
  property="mediaStoreSelectors">
<list>
  <bean class="hox.corem.server.media.ConditionalMediaStoreSelector">
    <property name="storeName" value="videostore"/>
    <property name="condition">
      <bean class="hox.corem.server.media.OrCondition">
        <property name="conditions">
          <list>
            <bean class="hox.corem.server.media.MatchCondition">
              <property name="primaryType" value="image"/>
            </bean>
            <bean class="hox.corem.server.media.MatchCondition">
              <property name="primaryType" value="video"/>
            </bean>
            <bean class="hox.corem.server.media.MatchCondition">
              <property name="primaryType" value="audio"/>
            </bean>
            <bean class="hox.corem.server.media.MatchCondition">
              <property name="primaryType" value="application"/>
              <property name="subType" value="vnd.rn-realmedia"/>
            </bean>
          </list>
        </property>
      </bean>
    </property>
  </bean>
</list>
</customize:prepend>
```

Example 3.7. Custom store configuration

For further details, see the Javadoc of package `hox.corem.server.media`.

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 when using Spring XML bean definitions.

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 "true" when at least one of the conditions is fulfilled.
<code>hox.corem.server.media.MatchCondition</code>	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 <code>MatchCondition</code> can take the following values: <code>primaryType</code> - Required primary MIME type of the content.

Condition Name	Description
	• <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>maxLength</i> - Required maximum content length in bytes. • <i>documentTypeName</i> - Required name of the content type of the target content item. Subtypes do not match. • <i>declaringDocumentTypeName</i> - Required name of the exact content type declaring the property that will receive the blob. Subtypes match only if the property was not redefined in a more specific type. • <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. It returns "true" when all values match.

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

<property>

Child elements: <bean>, <list>

Parent elements: <bean>

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

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

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

Table 3.3. Attributes of the property element

<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.

Configuring store locations for file storage

You can change the default locations for the storage of blobs in the file system by using Spring application properties.

- 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.

3.4 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.

3.5 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 GlassFish 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.iiop.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.

3.6 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 *MasterLive 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 *Blueprint Developer Manual* for more information.

CAUTION

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.



3.7 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 Content Server start

Before you start the *Content Server* for the first time, make sure that the content type definition file [`<xyz>-doctypes.xml`] can be found at the location configured with the parameter `cap.server.documentTypes`. Normally this file is placed in the directory `config/contentserver/doctypes`. 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 `<DB name>.properties` file of your database before the first start of the *Content Server*. See [Section 3.2.1, “Specifying Tablespace for the Content Server” \[33\]](#) for details.

Start *Content Server* Spring Boot application.

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` until the content of the *Master Live Server* has been replicated with the exception a few very recent changes.

Start the *Workflow Server* Spring Boot application.

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`.

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

Prerequisites

*Start w Content Management Server
Check for proper startup*

Start the Replication Live Server

*Start the Workflow Server
Upload workflows*

security of the system. Go to the User Management window in the editor for this purpose [see [Section 3.15, "User Administration" \[21\]](#)]. Make sure to update all relevant configuration files.

Later Content Server starts

Later starts of the *Content Server* 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.

By setting `replicator.force-online-switch=true` you can make sure that the *Replication Live Server* goes online immediately, regardless of the replication state and the *Master Live Server* availability. This can help to ensure that a failed *Master Live Server* cannot bring down the entire delivery environment, if you do not have any monitoring solution or watchdogs in place. It might, however, cause different CAEs to deliver different states of the content.

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.

3.8 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 3.9, "Administering Replication Live Servers" \[60\]](#) but you should also read [Section 3.8.1, "Backup Strategy" \[55\]](#). The recovery strategy depends on the server whose database has been corrupted.

3.8.1 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*.

CAUTION

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 3.3, "Configuring Blob Storage" \[43\]](#) 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` 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.SQLiteStore.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.

3.8.2 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.

3.8.3 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 3.9.4, "Restoring from Replication Live Server Backup" \[62\]](#).

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" \[200\]](#)] 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_re  
mote';
```

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 `$INSTALL_DIR/etc/keys/<database name>.<dbuser>.rijndael` has to be copied to the respective directory on

the *Master Live Server* [`<databasename>` 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 prop  
erty='replicator_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* (deprecated for editorial work) to query all publication events in the specific period.

3.9 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*

NOTE

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.



3.9.1 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 with the IOR URL of the *Master Live Server*:

```
replicator.publicationIorUrl=http://<MasterLiveServer
ComputerName>:<port>/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 Property Reference" \[278\]](#).
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`.

CAUTION

The setup of a *Replication Live Server* must be completed without additional restarts 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 *Replication Live Server* database has to be build from a backup of the *Master Live Server* database as described later in the manual.



3.9.2 Installing further Replication Live Servers

To install further *Replication Live Servers* you can repeat the instructions from [Section 3.9.1, “Installing the First Replication Live Server” \[60\]](#) 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 with the IOR URL of the *Master Live Server*:

```
replicator.publicationIorUrl=http://<MasterLiveServer
ComputerName>:<port>/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 4.2.5, “Properties for Replicator Configuration”](#) in *Deployment Manual*) remains unprocessed, the *Replication Live Server* will go `online`.

3.9.3 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.

3.9.4 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*.

3.9.5 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:

CAUTION

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.

NOTE

When you dump the database content you will get Workflow Server view tables which lead to problems during the replay of the dump. Since these tables are not required on a *Replication Live Server* you can exclude them from the dump. The table names are:

- mls.WFVPISates
- mls.WfVTISates
- mls.WfVTokens

With the MySQL *dump* tool, for example, you can use the parameters `--ignore-table`, to exclude specific tables.



4. If the passwords on the *Master Live Server* are encrypted with `cm encryptpass words`, the Rijndael key of the *Master Live Server* in the file `$INSTALL_DIR/etc/keys/<dbname>.<dbuser>.rijndael` has to be copied to the respective directory on the *Replication Live Servers* [`<dbname>` 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:core-media:///cap/content/701282:add" are harmless. They only say, that this content has been fed before.

3.9.6 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.

One way to disconnect the *Replication Live Servers* from *Master Live Servers* is by sending a POST HTTP request to `http://host:port/actuator/replicator` with an `application/json` body like `{"enable": false}`.

```
curl --request POST \  
  --url http://localhost:42081/actuator/replicator \  
  --header 'Content-Type: application/json' \  
  --data '{"enable": false}'
```

By setting the flag to false the service will be stopped but the application will be still up. All data changes are no more propagated from *Master Live Servers* until it is activated again. The service state can be by reverted by changing the value `{"enable": true}` inside the request body.

3.9.7 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.

An alternative way to check the replication service status is by querying the actuator **HTTP** endpoint as follows:

```
curl -X GET http://localhost:42081/actuator/replicator
```

Checking the Replicator Configuration

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, 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, which ever 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;
- `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;
- `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 on line`: The replication was started and has just replicated another complete public-

ation, 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.

3.10 Administrating Multi-Master Publishing

3.10.1 Enabling Multi-Master Publishing

You have to enable multi-master publication before the first start of the *Content Management Server*. To do so, you have to configure the following property of your *Content Management Server*.

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

CAUTION

It is not possible to switch from an existing single-master system to a multi-master system by simply setting `cap.server.multipleLiveServers=true`. Please read [Section 3.10.4, “Migrate to Multi-Master Management Extension” \[74\]](#) for a detailed description of the migration process.



3.10.2 Configuring Multi-Master Publishing

Configuration of Multi-Master publishing means configuration of the publication targets. For each target the *Content Management Server* needs a set of `publisher.target`. You have to replace `<n>` with a consecutive number for each property set, starting with 0.

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> .

Property	Description
<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 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>	The property <code>folders</code> 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: <pre>publisher.target[1].folders=internet,download,5173</pre> 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>.

Table 3.4. Properties used to configure Multi-Master Management

WARNING

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.



3.10.3 Adding Publication Targets

Publication targets can be added dynamically as needed. Simply install a new *Master Live Server* and add a new group of `publisher.target` to the *Content Management Server*.

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**.

3.10.4 Migrate to Multi-Master Management Extension

The wish to perform a migration project from single-master to multi-master 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-master 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 Multi-Master 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" \[79\]](#). 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" \[80\]](#) 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-master 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-master mode by verifying that the property `cap.server.multipleLiveServers` 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-master enabled.
8. Change the property `cap.server.multipleLiveServers` to true.
9. Start the *Content Management Server*.
10. Update the `publisher.target` 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" \[76\]](#)). 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 Site Manager* 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-master 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.

CAUTION

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-master mode by verifying that the property `cap.server.multipleLiveServers` 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` to true.
19. Start the *Content Management Server*.
20. Update the `publisher.target` 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-master 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-Master 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" \[77\]](#), 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" \[77\]](#), 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-master 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-master 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” \[77\]](#), 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 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.target` 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 con-

siderable 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" \[77\]](#) can be bridged by setting up a web server delivering a set of static pregenerated pages, if that is needed.

3.11 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` and `ObservedValueChangelog` tables. That is, because the `sequenceno` in these tables is a foreign key to the `sequenceno` in the `ChangeLog` table and therefore useless if you delete the respective row in the `ChangeLog` table.

CAUTION

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>
```

4. Delete also all but the last 100000 entries from the `ObservedValueChangeLog` table.

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

All but the last 100000 entries are deleted from the `ChangeLog`, `LinkChangeLog` and `ObservedValueChangeLog` tables.

3.12 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.UserProvider`. The following subsection explains how UserProviders are configured.

3.12.1 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 a JAAS configuration file, which is typically named `jaas.conf`. The location of the `jaas.conf` file is configured by the key `cap.server.login.authentication`. The value may be:

- A file path relative to `$COREM_HOME`
- An absolute file path
- A URL, especially a `classpath` URL

LoginModule Configuration in `jaas.conf`

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

```
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=
    (publisher|studio|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,studio.regex=studio"

    /* 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=.*,
    importer.regex=.*,system.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=.*,
    importer.regex=.*,system.regex=.*"
;
*/
};
```

Example 3.8. The `jaas.conf` file

NOTE

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 3.9. 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” \[88\]](#). 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

LdapLoginModule Options	Description
<i>port</i>	LDAP server port number. If you want to use LDAP over SSL switch to 636, which is the default port for SSL connection.
<i>domain</i>	Domain to serve
<i>protocol</i>	The protocol to use. Change to "ssl" if you want to use LDAP over SSL.

Table 3.6. Options of the LdapLoginModule

For further LDAP over SSL configuration see [Section 3.12.5, “Connecting LDAP Over SSL” \[98\]](#).

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.

Service	Applications
debug	Applications used for debugging like <code>cm_dump</code>
editor	Editor applications like <i>Site Manager</i>
importer	<i>CoreMedia Importer</i>
publisher	Publisher in <i>contentserver</i>
replicator	Replicator in <i>CoreMedia Replication Live Server</i>
studio	Studio Server, User Changes App
system	System applications like <code>cm_documentcollector</code>
webserver	<i>CAE, Elastic Worker</i>

Service	Applications
workflow	<i>CoreMedia Workflow Server</i>
feeder	<i>Content Feeder, CAE Feeder</i>

Table 3.7. CoreMedia services

When users, no matter if built-in or LDAP user, log on to the *Content Server*, they need 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

Note: An administrator can manually query and remove used licenses from the database table. See the server tool `cm usedlicenses` in [Section "Usedlicenses" \[190\]](#).



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 `LdapLoginModule` can be configured 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.
<i>depth</i>	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 or-ed. If one of the values matches the regular expression, the predicate returns true. 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.
<code><service>.regex</code>	<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

NameLoginPredicate Options	Description
	regular expression are allowed to log on. <service> must be one of "debug", "editor", "importer", "publisher", "replicator", "studio", "system", "webserver", "workflow", or "feeder".
<i>regex</i>	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.

Table 3.8. NameLoginPredicate options

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
<i>negative</i>	Same as for NameLoginPredicate
<i>depth</i>	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.corem.usermanager.User#getAttributeKeys()</code>. For the LDAP server, these are the values configured with the properties <code>cap.server.userproviders[#].ldap.user.custom-attributes</code> for a user and <code>cap.server.userproviders[#].ldap.group.custom-attributes</code> for a group in the contentserver application properties. The values are logically or-ed. If one of the values matches the regular expression, the predicate returns true. 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.
<i><service>.regex</i>	Same as for NameLoginPredicate

AttributeLoginPredicate Options	Description
<code>regex</code>	Same as for NameLoginPredicate

Table 3.9. AttributeLoginPredicate options

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>	The JndiNameLoginPredicate 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. For example, with depth=0 only the JNDI name of the user is used. With depth=1 the JNDI names of the user and the JNDI names of his direct groups are matched against the regular expression.
<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=
(publisher|studio|watchdog|workflow|
webserver|importer) "

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

Example 3.10. jaas.conf example

The result of matching the regular expression `(publisher|studio|watchdog|workflow|webserver|importer)` is negated for users requesting the editor service. This means that the built-in users with the names `publisher`, `studio`, `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 *Content Server* 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 *Content Server* log after a user has tried to log on. You do not need to restart the *Content Server* when changing the log level.

3.12.2 Configuration of UserProviders

Here is a minimal example configuration for an Active Directory Server `UserProvider`:

```
cap.server.userproviders[0].provider-class=\
com.coremedia.ldap.ad.ActiveDirectoryUserProvider
cap.server.userproviders[0].java.naming.security.principal=\
CN=Administrator,CN=Users,DC=acme,DC=com
cap.server.userproviders[0].java.naming.security.credentials=secret
cap.server.userproviders[0].ldap.host=activedirectory.acme.com
cap.server.userproviders[0].ldap.base-distinguished-names[0]=\
CN=Users,DC=acme,DC=com
```

Example 3.11. Example for Active Directory Server

The first property configures the `UserProvider` class. The other properties denote the account to use for connections, the hostname of the Active Directory Server, and the distinguished names where to search for users and groups. Further distinguished

names can be added by incrementing the index value of `base-distinguished-names`.

Further providers can be added by incrementing the index value of `userproviders`. The following example shows the configuration of another `UserProvider` of a custom class with some custom properties:

```
cap.server.userproviders[1].provider-class=com.acme.MyUserProvider
cap.server.userproviders[1].properties[com.acme.a-configuration-property]=foo
cap.server.userproviders[1].properties[com.acme.another-property]=bar
```

Example 3.12. Configuration of a second UserProvider

The following sections introduce some predefined user providers classes.

3.12.3 LdapUserProvider

The class `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 but can be subclassed by concrete provider classes. It has some configuration options (see `LdapUserProviderConfigurationProperties`), many of which are specific to concrete subclasses, like particular LDAP attributes for names and IDs. Subclasses can override `com.coremedia.ldap.LdapUserProvider.completeJndiConfiguration` and `com.coremedia.ldap.LdapUserProvider.completeLdapConfiguration` to complete the configuration with reasonable default values in order to allow for short configurations. Ideally, only a few environmental properties like the LDAP hostname and the account to use need to be configured, like in the above example for an `ActiveDirectoryUserProvider`.

All properties which start with `java.naming` are evaluated by the JNDI framework. Some mandatory `java.naming` properties, like `java.naming.security.principal` and `java.naming.security.credentials` are covered by the `JndiConfigurationProperties`, so that you can configure them directly as Spring configuration properties. These properties are sufficient for most JNDI environments. If you need other `java.naming` properties in your environment, though, you can add them to the generic Map-valued `cap.server.userproviders.properties` property, and the `LdapUserProvider` propagates them to the JNDI environment.

```
cap.server.userproviders[0].java.naming.security.principal=\
CN=Rudy_ReadOnly,CN=Users,DC=acme,DC=com
cap.server.userproviders[0].java.naming.security.credentials=123456
```

```
cap.server.userproviders[0].properties[java.naming.language]=en
```

Example 3.13. Configuration of java.naming properties

The last mandatory environmental property denotes the distinguished names of the LDAP subtrees to search for users and groups. In the easiest case, you simply configure a single high-level subtree. In more complex LDAP directory layouts, you might want to restrict the CMS access more fine grained to particular subtrees. The following example configures the `CN=Users` subtree of the domain as base distinguished name, which is suitable for the default ActiveDirectory structure.

```
cap.server.userproviders[0].ldap.base-distinguished-names[0]=\
CN=Users,DC=acme,DC=com
```

Example 3.14. Base Distinguished Names

CAUTION

Due to a bug in the JNDI implementation since Java 8 CoreMedia's LDAP integration cannot correctly cope with referrals. If you use Active Directory with Trust Relationships, you can only use DNS 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]
```



If you cannot express the search scope for users and groups in terms of distinguished names, you may specify two sorts of filters: OU filters (`LdapUserProviderConfigurationProperties.Ou#filter`), which operate on the LDAP level, and member filters (`LdapUserProviderConfigurationProperties#memberFilter`), which operate on the Java level. Both have negative drawbacks (see their Javadoc for details), though, so you should try to get along with distinguished names only.

3.12.4 ActiveDirectoryUserProvider

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

NOTE

If you are migrating from an earlier version of *CoreMedia Content Cloud*, you will have noticed that the `SimpleActiveDirectoryUserProvider` is new. The `ActiveDirectoryUserProvider` is still available, no action is required for existing projects, and the following configuration steps are the same for both UserProviders.

The `ActiveDirectoryUserProvider` works only with Windows Server Active Directory, while the `SimpleActiveDirectoryUserProvider` is also suitable for Azure Domain Services. If the `userPrincipalName` attribute is equivalent to the `sAMAccountName` and the distinguished name of users, the UserProviders are compatible. While this is the case in Windows Server Active Directory with default configuration, it does not hold for Azure Domain Services. Therefore, CoreMedia introduces the new `sAMAccountName` based `SimpleActiveDirectoryUserProvider`, because the `userPrincipalName` in Azure Domain Services is not suitable for our needs.

Our recommendation is to use the `SimpleActiveDirectoryUserProvider` in new projects and to use the `ActiveDirectoryUserProvider`:

- In existing projects (in order to avoid any risk).
- If you definitely favor the `userPrincipalName` over the `sAMAccountName`.

1. Tell the *Content Server* to use an Active Directory Server for authentication by configuring the following properties. (If you configure multiple UserProviders, take care for the grouping numbers in the property keys.)

```
cap.server.userproviders[0].provider-class=\
com.coremedia.ldap.ad.SimpleActiveDirectoryUserProvider
```

2. Set the environment specific Active Directory Server properties as follows:
 - a. Set your Active Directory Servers host (and port, if it deviates from the standard ports 389 or 636 for LDAPS):



```
cap.server.userproviders[0].ldap.host=<your-active-directory-server-host>
```

- b. Set your Administrator's distinguished name and password:

```
cap.server.userproviders[0].java.naming.security.principal=\
CN=Administrator,CN=Users,DC=your,DC=domain
cap.server.userproviders[0].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 distinguished name by entries of increasing index for `base-distinguished-names` (see also [Section 3.12.3, "LdapUserProvider" \[94\]](#)).

```
cap.server.userproviders[0].ldap.base-distinguished-names[0]=\
CN=Users,DC=your,DC=domain
```

3. Activate the `hox.corem.login.LdapLoginModule` in `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.

NOTE

The above description applies to Windows Server 2008 and newer. If you use an Azure Domain Service instead, the default location of users and groups is `OU=AADDCUsers` rather than `CN=Users`. This affects steps 2.b and 2.c.



If you are using an Azure Domain Service or if your Windows Server Active Directory is restricted to LDAPS, proceed with the next section, "Connecting LDAP Over SSL".

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`.

3.12.5 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. Add the following property in the contentserver application properties:

```
cap.server.userproviders[0].java.naming.security.protocol=ssl
```

3. Configure the `hox.corem.login.LdapLoginModule` in the file `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.

3.12.6 Common Customizations

The Active Directory LDAP schema has some redundancy. User names can be derived from various attributes, like `sAMAccountName`, `userPrincipalName` or `mail`. User domains can be derived from `userPrincipalName` or from the distinguished name. In a default Active Directory setup all that data is consistent, but in principle it may be ambiguous. In the latter case, you must decide which attributes are appropriate to reflect your users in *CoreMedia Content Cloud*.

CoreMedia Content Cloud provides two `UserProviders` for Active Directory to start over with:

- The `SimpleActiveDirectoryUserProvider` derives users' names from the `sAMAccountName` and domains from the DC components of the distinguished names. This is suitable for most Active Directory setups, Windows Server as well as Azure Domain Services. If you have no particular requirements with regards to user names and domains, you should start over with the `SimpleActiveDirectoryUserProvider`.
- The `ActiveDirectoryUserProvider` derives both, users' names and domains, from the `userPrincipalName` attribute, whose value has the format "name@domain".

Simple User Name Attributes

The user name attribute determines which `UserProvider` you should extend in order to implement your customizations. If you use `sAMAccountName` (which is recommended), you can simply start over with the `SimpleActiveDirectoryUserProvider`. If you want to use another attribute that contains exactly the user's name, for example `givenName`, you can also use the `SimpleActiveDirectoryUserProvider` and configure the attribute name:

```
cap.server.userproviders[0].ldap.user.filter=(&(objectClass=user)(givenName=*))
cap.server.userproviders[0].ldap.user.name-attribute=givenName
```

The `UserProviders` fetch only the LDAP attributes they need. If you use additional attributes, like `givenName`, you must configure them as `cap.server.userproviders[0].ldap.user.attributes`.

```
cap.server.userproviders[0].ldap.user.attributes[0]=givenName
```

Combined Name@Domain Attributes

If you prefer the `userPrincipalName`, you should start with the `ActiveDirectoryUserProvider`, which supports the attribute format "name@domain". The domain from the attribute may differ from the domain of the distinguished name. In order to resolve this ambiguity, you must specify the `UserProvider`'s domains explicitly to override the automatic computation from the configured base distinguished names:

```
cap.server.userproviders[0].ldap.domains[0]=example.org
cap.server.userproviders[0].ldap.domains[1]=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 base distinguished names.

For another attribute of the same format (like `mail`) configure the attribute name:

```
cap.server.userproviders[0].ldap.user.filter=(&(objectClass=user)(mail=*))
cap.server.userproviders[0].ldap.user.name-attribute=mail
cap.server.userproviders[0].ldap.user.domain-attribute=mail
cap.server.userproviders[0].ldap.user.attributes[0]=mail
```

3.12.7 LDAP User and User Changes Application

The User Changes Spring Boot application tracks the changes a user has made in the content repository. Therefore, a user can, for example, start a publication workflow with all recently changed contents. However, some contents should be excluded from content

tracking. Especially the home folder of a user or some system folders, which usually contain only configuration data rather than editorial content.

By default, the home folders follow the `/Home/* /MyPreferences` pattern. When you use an LDAP user provider, the path to the `MyPreferences` folder (and to other folders) might be different, because the home folders of the Active Directory users represent the OU structure in the AD. Therefore, you have to add paths with a corresponding depth. Adapt the configuration as follows:

Adapt paths to OU structure

1. In the Blueprint workspace add a `userchanges.paths.exclude` property to the properties file `component-user-changes-blueprint.properties` in the `apps/user-changes/modules/server/user-changes-blueprint-component` module below `src/main/resources/META-INF/coremedia/`. Create a new file if non-existent.

For a Docker deployment, add the property to `apps\user-changes\docker\user-changes\src\docker\config\application.properties`.

2. Add the default entries to the property:

```
/Home/*/EditorPreferences,/Home/*/My Preferences,/Home/*/My Dictionary,/System/Public Dictionary
```

3. Add your own paths to the `EditorPreferences`, `My Preferences` and `My Dictionary` directories to the property. Add wildcard elements matching the users' OUs, for example:

```
/Home/*/*/* /My Preferences
```

When you now build the Blueprint workspace or the Studio Spring Boot application, the paths will be excluded for the User Changes Spring Boot application.

3.13 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, 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`. That implies, that most of the tools work only on one server simultaneously.

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)

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

Table 3.11. Common options of server utilities

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.

The password parameter `-p` is always optional. You can provide the password as the environment variable `REPOSITORY_PASSWORD` instead. If you don't specify the password at all, the tool will prompt for it at runtime.

Protecting Passwords

A password that is provided via the command line might be visible in the shell history or in the list of running processes. A password provided via an environment variable might accidentally be passed to other commands, if the environment variable is set in a long-running shell or even during shell initialization.

If you must not or cannot enter the password manually, consider setting the environment variable in a wrapper script. Preferably the password is retrieved automatically from a secure password vault. If the password is stored in the shell script, the shell script must not be readable by other users.



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.

3.13.1 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, ...)
- `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-link-type`: Validates that links in existing content are valid with regard to the `LinkType` of the document type definition
- `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> ... |
-U <uuid1> <uuid2> ... |
-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 or UUIDs, 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
-U,--uuids                      UUIDs of contents 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
<code>-f <file name></code>	Name of the file where to dump the output using UTF-8 encoding. By default, the output is written to stdout.

Parameter	Description
-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
-U <UUIDs>	UUIDs of contents to dump.
-t <paths>	The paths of resources to dump.

Table 3.12. Options of 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
  uuid: c4ca4238-a0b9-3382-8dcc-509a6f75849b
  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, or their UUIDs with `-U` option, for example,

```
> cm dump -u admin -p admin -t /work/article1 -t /work/image1
> cm dump -u admin -p admin -U a935da47-271b-4712-9612-acf8aa1b965e
```

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`.

As alternative to this, you may use the option `-U` to dump a user or group by its UUID, just as you can do it for content UUIDs.

```
> cm dump -u admin -p admin -U c4b75f33-b75f-3b23-9e13-bbfbf072f1c4d
```

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” \[11\]](#) 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.

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 Section 4.4, “Communication between the System Applications” in *Operations Basics*).

The program is executed with:

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

Example:

```
> bin/cm ior `wget -qO- http://localhost:40180/ior`

...
IOR:0000000000000002549444c3a686f782f636f72656d2f636f7262612
f4c6f67696e536572766963653a312e300000000000000010000000000
0000340001000000000016686f78706332342e636f72656d656469612e6
36f6d00042800000010000000003ab08a890005c0a800000000
IOR<IDL:hox/corem/corba/LoginService:1.0><IIOP:1.0:pcname.
coremedia.com:1064>
```

Example 3.15. 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 of license

```
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
```

The `license` tool has only one additional option:

Parameter	Description
-v	Verbose output, prints out additional information

Table 3.13. 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/studio, 0/15c, 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.

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* as well as the standard administrative tool are not included. Therefore, only infrastructure processes (servers, 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 of processorusage

```
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)
-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
```

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.

Table 3.14. The parameters of processorusage

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:

Parameter	Description
<code>-v</code>	Verbose output, prints out additional information

Table 3.15. Options of the `publications` utility

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” \[103\]](#)).

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

# 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
```

Example 3.16. Result of `repositorystatistics`

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 <ior url> url to connect to
```

Usage of the rules utility

```
-t, --paths <path>    path of contents to dump
-v, --verbose           enables verbose output
```

`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 3.16. 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 3.15.2, “User Rights Management” \[217\]](#).

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
<code>-v</code>	Verbose output, prints out additional information

Table 3.17. 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 of systeminfo

```
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
```

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 Link Type

The tool `validate-link-type` can be used to check if links in existing content point to documents of the correct type as declared with the `LinkType` attribute in the document type definition. It can also be used to check in advance whether the `LinkType` can be changed to a different possibly more specific type without causing issues for existing content. The purpose of the tool is to help during schema update. For details, see [Section "Changing LinkType of LinkListProperty" \[273\]](#).

Usage of validate-link-type

```
cm validate-link-type {-u user} [-p password] {-t content type}
{-l link property} [-a] [-n]
```

Parameter	Description
<code>{-t --content-type} content-type</code>	The name of the content type to check links in.
<code>{-l --link-property} property</code>	The name of the link list property in the specified content type.

Parameter	Description
<code>-n --new-link-type content-type</code>	New link type to use as expected target content type for the link list property instead of the <code>LinkType</code> from the document type definition. Use it to check in advance if the <code>LinkType</code> attribute can be changed in the document type definition.
<code>-a --all-versions</code>	Check links in all document versions, instead of just checking the latest version.

Table 3.18. Parameters of `validate-link-type`

Example

For example, use the following command to check whether the `master` property of all `Article` document versions only links to documents of the type as declared in the document type definition:

```
cm validate-link-type -u admin -t Article -l master -a
```

If no invalid links are found, the tool will output:

```
No links with wrong type found.
```

In another example, let's imagine you are thinking about changing the `LinkType` attribute for property `pictures` of type `Teasable` from `Media` to subtype `Picture`. In that case, you should first check whether such a restriction would be violated by existing content in current or old versions:

```
cm validate-link-type -u admin -t Teasable -l pictures -n Picture -a
```

In this example, the tool reports two links that would violate the new link type `Picture`. In this example, it's the same link in an old and in the latest version of the same content. If option `-a` (`--all-versions`) were omitted, only the link in the latest version would have been reported.

```
Checking 2654 contents. This may take some time...
Found link of wrong type: coremedia:///cap/version/4638/2 ->
coremedia:///cap/content/5134
Found link of wrong type: coremedia:///cap/version/4638/3 ->
coremedia:///cap/content/5134
```

Note, that the tool outputs an additional line here at the start to warn about its expected runtime. If you intend to use the output in a script, make sure to ignore all rows that don't start with `"Found link of wrong type:"`.

Validate Multi-Site

The tool `validate-multisite` helps you to manage multiple localized variants of contents in *CoreMedia Content Cloud* as described in chapter "CoreMedia Content Cloud Website and Content Structure/Localized Content Management" of the [Blueprint 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] [-t] [other...]
```

Example 3.17. Usage of `validate-multisite`

Parameter	Description
<code>{ -b --below } <i>folder</i></code>	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.
<code>{ -f --file } <i>file</i></code>	Output file for writing the detected issues in tab-separated value format.
<code>-fow --fail-on-warning</code>	Fail with a non-zero exit code when a warning is generated.
<code>-t --thorough</code>	Disable fail early approach and search for failures more thorough.

Table 3.19. Parameters of `validate-multisite`

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" \[115\]](#).

Issues

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	<code>SitesService</code> is not defined. None of the multi-site features will work.
Symptom(s)	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 <code>SitesService</code> . Typically, by adding a dependency to <code>com.coremedia.cms:cap-multisite</code> and importing <code>multisite-services.xml</code> .

```
<import
resource="classpath:/com/coremedia/cap/multisite/multisite-services.xml"/>
```

MS-VALIDATION-1001 - Missing Site Model

Severity	ERROR
Description	SiteModel is not available in SitesService.
Symptom(s)	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 com.coremedia.cms:cap-multisite, importing multisite-services.xml and adding your configured SiteModel bean. <import resource="classpath:/com/coremedia/cap/multisite/multisite-services.xml"/> <customize:replace id="addYourSiteModel" bean="sitesService" property="siteModel" custom-ref="yourSiteModel"/>

It is recommended to create a bean of the type com.coremedia.cap.multisite.DefaultSiteModel or of a custom subtype.

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).
Symptom(s)	Some multi-site features may not work or may cause unexpected states depending on the affected property.
File Output	3. name of property
Option(s)	Set the required/recommended property value.

MS-VALIDATION-2001 - Deprecated Property

Severity	INFO
Description	The property is deprecated.
Symptom(s)	<i>none</i>
File Output	3. deprecated property name 4. current value of property
Option(s)	Consult deprecation notices for details and possible replacement.

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.
Symptom(s)	Sites will not be available.
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.
Symptom(s)	Some, if not all multi-site features will not work or cause unexpected states.
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.
Symptom(s)	Some, if not all multi-site features will not work or cause unexpected states.
File Output	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.
Symptom(s)	You will always have to publish the master content prior to its derived contents, and you cannot withdraw the master content without withdrawing all derived contents first.
File Output	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.
Symptom[s]	Most likely none, as a fallback exists. For example if the locale property is missing, the typical fallback is to the site indicator's locale.
File Output	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.
Symptom[s]	Some, if not all multi-site features will not work or cause unexpected states.
File Output	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-3006 - Invalid Master Link Type	
Severity	WARN, INFO
Description	A content type having a master link property of wrong type. It is expected that the type of the master link is just the same as the content type defining it. If the link type is a parent type this issue is considered INFO level as it does not break processing. If the link type is of any other type, this issue is considered WARN level as it might cause severe problems during multi-site processing.
Symptom[s]	Most likely none, if the link type is compatible to the current content type. May break multi-site features, though, if an editor sets a link to a content different to current type. If the link type is incompatible, multi-site feature may or will break, like translation processes may escalate.

File Output	3. content type 4. property 5. actual link type
Option[s]	Adjust your content type model so that the master link is always of the same type as the current content type. For details, see Section "Changing LinkType of LinkListProperty" [273] .
MS-VALIDATION-4000 - Invalid Property Value	
Severity	WARN
Description	Property value is invalid.
Symptom[s]	Multi-site features may produce unexpected results. If, for example, the locale is not idempotent regarding <code>Locale.forLanguageTag(locale).toLanguageTag()</code> multi-site features may evaluate to different locales.
File Output	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.
Symptom[s]	Multi-site features may produce unexpected results. If for example multiple site root documents got linked, retrieving the site root document may cause arbitrary results.
File Output	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.
Symptom(s)	When deriving a site, the derived sites may not be created as sibling of the current site.
File Output	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.
Symptom(s)	The denoted site root folder may be wrong, and may result in unexpected contents to be contained or not to be contained in site. When deriving a site, the derived sites may not be created as sibling of the current site.
File Output	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.

Symptom(s)	Sites will not be available.
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.
Symptom(s)	Ambiguous results when determining site/site properties of a given content. Multi-site features such as translation may cause unexpected results.
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.
Symptom(s)	Some sites will not be available to be selected as preferred site. Ambiguous results when determining site/site properties of a given content. Multi-site features such as translation may cause unexpected results.
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.
Symptom[s]	Site represented by this site indicator will not be available (site will be considered destroyed).
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.
Symptom[s]	When deriving a site, the site root document will not be derived. Instead, the site indicator of the derived site will direct to the same site root document as its master 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-4009 - Discouraged Property Value	
Severity	INFO
Description	The value of the property is discouraged due to known disadvantages.

Symptom[s]	<i>none</i>
File Output	3. property name 4. current value of property
Option[s]	Consult documentation and release notes for the recommended value and migration efforts.

MS-VALIDATION-4020 - Missing Translation Settings Content Type

Severity	ERROR
Description	Content type defined in property <code>translationSettingsDocumentType</code> of site model does not exist.
Symptom[s]	Content synchronization will not work.
File Output	3. referenced non-existing content type

MS-VALIDATION-4021 - Missing Translation Settings Property Value

Severity	ERROR
Description	The denoted property value of a translation settings document is empty or unset but is required.
Symptom[s]	Content synchronization will not work.
File Output	3. affected translation settings document's id 4. name of property
Option[s]	Set the required property value.

MS-VALIDATION-4022 - Invalid Translation Settings Property Value

Severity	ERROR
Description	The denoted property value of a translation settings document is invalid.

Symptom[s]	Content synchronization will not work.
File Output	3. affected translation settings document's id 4. name of property
Option[s]	Set the required property value.
MS-VALIDATION-5000 - Type not Localizable	
Severity	WARN
Description	A content inside a site is not of a localizable content type.
Symptom[s]	This might cause issues in translation process, such as that contents cannot be marked having reached a new translation state.
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.
Symptom[s]	When deriving a site, or propagating the content to a derived site, the locale will not be adjusted. This behavior will break as soon as a site in the site hierarchy shares the same locale as the content. In lower levels the site locale will be adapted to the site locale which may be unexpected.
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.
Symptom(s)	None, because the locale of the content will default 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.
Symptom(s)	There are no immediately visible symptoms. May cause problems on delivery though, for example when displaying language toggles.
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.
Symptom(s)	Most likely none, but content may be recreated unexpectedly during translation processes, if it gets linked in master content.

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	• MS-VALIDATION-6001 - Content not Used in Derived Site but some Content Exists [127] • MS-VALIDATION-6002 - Content without Master [127]

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.
Symptom[s]	Possible counterpart will not receive any translation results. Instead, if the master content gets translated, a new derived content will be created with a non-conflicting name.
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	• MS-VALIDATION-6000 - Content not Used in Derived Site [126] • MS-VALIDATION-6003 - Content without Master but some Content Exists [128]

MS-VALIDATION-6002 - Content without Master

Severity	INFO
----------	------

Description	Counterpart to MS-VALIDATION-6000 [126] : 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.
Symptom(s)	<i>none</i>
File Output	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	• MS-VALIDATION-6000 - Content not Used in Derived Site [126] • MS-VALIDATION-6003 - Content without Master but some Content Exists [128]

MS-VALIDATION-6003 - Content without Master but some Content Exists

Severity	INFO
Description	Counterpart to MS-VALIDATION-6001 [127] : 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.
Symptom(s)	Possible counterpart will not receive any translation results from master. Instead, if the master content gets translated, a new derived content will be created with a non-conflicting name.
File Output	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	• MS-VALIDATION-6001 - Content not Used in Derived Site but some Content Exists [127] • MS-VALIDATION-6002 - Content without Master [127]

MS-VALIDATION-6004 - Root Content with Master

Severity	INFO
----------	------

Description	A content in a root site defines a master.
Symptom(s)	Most likely none, but may signal that either the site indicator misses a master link. Otherwise, the master link is just ignored.
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 belonging neither to the master site, nor the content's site.
Symptom(s)	Content will never receive updates from the denoted master, as propagation of localization is organized by site, not content hierarchies.
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.
Symptom(s)	Only one arbitrary derived content will receive updates from master. The algorithm may even choose different derived contents on each update process.
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.

Symptom[s] Most obvious symptom is, that auto-merge properties will not receive any updates. Linklists for example will not receive newly inserted links in master content.

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

- [MS-VALIDATION-7001 - Master Version Invalid \[130\]](#)
- [MS-VALIDATION-7002 - Master Version not Localized yet \[131\]](#)
- [MS-VALIDATION-7003 - Master Version not up to Date \[131\]](#)
- [MS-VALIDATION-7004 - Master Version not set \[132\]](#)

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.

Symptom[s] Most obvious symptom is, that auto-merge properties will not receive any updates. Linklists for example will not receive newly inserted links in master content.

File Output

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	• MS-VALIDATION-7000 - Master Version Destroyed [130] • MS-VALIDATION-7002 - Master Version not Localized yet [131] • MS-VALIDATION-7003 - Master Version not up to Date [131] • MS-VALIDATION-7004 - Master Version not set [132]

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.
Symptom[s]	Content will be marked as not localized yet.
File Output	3. affected content id 4. master content id 5. denoted not yet localized 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 [130] • MS-VALIDATION-7001 - Master Version Invalid [130] • MS-VALIDATION-7003 - Master Version not up to Date [131] • MS-VALIDATION-7004 - Master Version not set [132]

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.
Symptom[s]	Content will be marked as not being up-to-date.

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 [130] • MS-VALIDATION-7001 - Master Version Invalid [130] • MS-VALIDATION-7002 - Master Version not Localized yet [131] • MS-VALIDATION-7003 - Master Version not up to Date [131]
----------	--

MS-VALIDATION-7004 - Master Version not set

Severity	WARN
----------	------

Description	Derived content has no master version.
-------------	--

Symptom[s]	Most obvious symptom is, that auto-merge properties will not receive any updates. Linklists for example will not receive newly inserted links in master content.
------------	--

File Output	3. affected content id
	4. master content id
	5. the latest master version

Option[s]	Set the master version of the derived content. If you choose a positive number, it signals, that the content received the latest translation from master (master and derived are up-to-date). If you choose a negative number, it signals, that the derived content got copied from the given master version and it is marked as not translated yet. Benefit of the positive number is, that there is nothing more to do. Benefit of the negative number is, that you may trigger an editorial review process for adjusted contents.
-----------	--

See Also	• MS-VALIDATION-7000 - Master Version Destroyed [130] • MS-VALIDATION-7001 - Master Version Invalid [130] • MS-VALIDATION-7002 - Master Version not Localized yet [131] • MS-VALIDATION-7003 - Master Version not up to Date [131]
----------	--

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.
Symptom(s)	Such external links will not be adjusted during localization processes (derive site, translation, ...). While this does no harm to the translation processes, it may be unexpected at delivery, as for example navigation contexts cannot be resolved.
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.
Symptom(s)	Most likely none regarding localization processes.
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 for strong links, INFO for weak links
Description	A content links to a deleted content.
Symptom(s)	Most likely none regarding localization processes. Note, though, that deleted links behave like external links (links to content outside of current site). I.e., the link will not be mapped upon localization processes. If for any reason the linked content is restored, you may end up with additional issues regarding multi-site validation and localization processes.
File Output	3. affected content id 4. link property containing link to deleted content

Option[s]	Adjust the link.
MS-VALIDATION-8003 - Link to Deleted Master	
Severity	WARN
Description	A content links to a deleted content in its master property.
Symptom[s]	Most likely none regarding localization processes. It may signal, that a content should be deleted in derived sites, too.
File Output	3. affected content id
Option[s]	Delete the derived content after deletion of the master content. Or adjust the link to point to the correct master content.

Table 3.20. Issues of validate-multisite

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.

3.13.2 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
- `jshell`: The *JShell* wrapper lets you interactively access the *Content Server* using the *Unified API*.
- `groovysh`: The *Groovy Shell* lets you interactively access the *Content Server* using the *Unified API*.
- `content-uuid-export`, `generate-content-uuid-map`, `content-uuid-export`: These tools will assist you synchronizing UUIDs in different environments, like development and production, for *Content Management Server* upgraded from before 2010.1 or for example as fallback solution for `serverexport`, `serverimport`, if you forgot enabling UUID-export.

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
- `encryptpasswordproperty`: Encrypt a given password for use in property files

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” \[108\]](#)].

```
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
```

Usage of cancelpublication

Changepassword

The `changepassword` tool changes the password of a user on the server to which it connects [`-url` property]. An administrator can change the password of every other user, all other users may only change their own passwords.

```
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
```

Example 3.18.

The options have the following meaning:

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.

Table 3.21. Options of the `changepassword` tool

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.

NOTE

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
```

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>yyyyMMddHHmms s</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>yyyyMMddHHmms s</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.

Table 3.22. Options of the clean recycle bin

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>
Based on given version selector class:
>>> Selector (default): DefaultVersionSelector <<<
[--selector-class
com.coremedia.cmdline.management.DefaultVersionSelector]
[--keep-number <n>]
[--keep-days <d>]
[--published-only | --unpublished-only]
[--approved-only | --unapproved-only]
>>> Custom Selector (deprecated): <<<
--selector-class <classname>
[--selector-param <name1:value1>]
And independent of any version selector:
[--simulate]
[--export]
[--directory <directory>]
[--export-to-subdirs <mode>]
[ -R ] [ -l <limit> ] [ <id1>... | -I <id1>... | -U <uuid1>... |
-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>
-kn,--keep-number <arg>	minimum Number of versions to keep <default=1>
-l,--limit <limit>	limits the number of content objects to select, negative for unlimited which is

<code>-lc,--selector-class <arg></code>	also the default
<code>-lp,--selector-param <arg></code>	VersionSelector implementation class
<code>-p,--password <password></code>	custom VersionSelector parameter
<code>-po,--published-only</code>	password for login
<code>-R,--recyclebin</code>	clean published versions only
<code>-s,--simulate</code>	include contents from recycle bin
	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>-U,--uuid <uuid></code>	UUID(s) to select contents
<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

General Options

Parameter	Description
<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>	Defaults to <code>com.coremedia.cmdline.management.DefaultVersionSelector</code> . The class name of a custom VersionSelector (either implementing <code>com.coremedia.cmdline.management.VersionSelector</code> or <code>com.coremedia.cmdline.management.ExtendedVersionSelector</code> or extending <code>com.coremedia.cmdline.manage</code>

Parameter	Description
	<code>ment.AbstractVersionSelector</code> (recommended)]. For more details how to implement custom selectors, see section below.
<code>--selector-param</code>	(deprecated) An optional parameter that will be passed to the custom <code>VersionSelector</code> as string in its constructor argument. For more details how to implement custom selectors, see section below.
<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 linked by OR. <code>Clean Versions</code> will traverse folders recursively.
<code>-R</code>	Content Selection Add contents from recycle bin. 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 linked by OR. <code>Clean Versions</code> will traverse folders recursively.

Parameter	Description
<code>-U <UUID></code>	Content Selection Select contents via UUID. This is an alternative if you want to mix UUIDs with for example content queries. Parameter can be used multiple times. Results will be linked by OR. <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 linked by OR. <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 linked by OR. <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.

Table 3.23. Switches of the version collector

Options of DefaultVersionSelector

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.
<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

Parameter	Description
	the latest published version will never be deleted by this tool. Will be ignored for custom version selectors.
<code>--approved-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>--unapproved-only</code>	

Table 3.24. Switches of the version collector

NOTE

For information about the UAPI query syntax see Section 5.7, “Query Service” in *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" (deprecated approach):

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

Example for calling *Clean Version* using a custom VersionSelector extending the default version selector:

```
cm cleanversions -u admin -p admin
--selector-class my.package.OverriddenDefaultVersionSelector
--keep-number 3
--my-custom-option true
```

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 named class must be a public class implementing `com.coremedia.cmdline.management.VersionSelector` with a no-args constructor. Deprecated usages with one argument of type `java.lang.String` are still supported. In the latter case, a `String` containing an additional parameter will be passed to the constructor (see `--selector-param`).

For enhanced parameter support it is recommended to implement `com.coremedia.cmdline.management.ExtendedVersionSelector` instead, or even better to extend `com.coremedia.cmdline.management.AbstractVersionSelector` instead. If you want to extend/customize the default version selector, you may also decide to override `com.coremedia.cmdline.management.DefaultVersionSelector` instead.

Having provided a custom `ExtendedVersionSelector` you have to register it to the service loader providing a file `META-INF/services/com.coremedia.cmdline.management.ExtendedVersionSelector` in your classpath mentioning the fully qualified name of your custom selector.

```

package my.package;

import com.coremedia.cap.content.Version;
import com.coremedia.cmdline.management.AbstractVersionSelector;

import java.util.Collections;
import java.util.List;

public class SelectOldestVersionSelector extends AbstractVersionSelector {
    @Override
    public List<Version> selectVersions(List<Version> versions) {
        return versions.stream()
            .findFirst()
            .map(Collections::singletonList)
            .orElseThrow(IllegalStateException::new);
    }
}

```

Example 3.19. Example of simple custom selector

```

package my.package;

import com.coremedia.cap.common.CapStructHelper;
import com.coremedia.cap.content.Version;
import com.coremedia.cmdline.management.AbstractVersionSelector;
import org.apache.commons.cli.CommandLine;
import org.apache.commons.cli.OptionBuilder;
import org.apache.commons.cli.Options;

import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class SkipArchivedVersionSelector extends AbstractVersionSelector {

    private static final String LONG_OPT = "skip-archived";
    private static final String SHORT_OPT = "sa";
    private static final String ARCHIVED_PROPERTY = "archived";
    private boolean skipArchived;

    @Override
    public void fillInOptions(Options options) {
        addOption(options, OptionBuilder
            .withDescription("only versions not marked as archived")
            .withLongOpt(LONG_OPT)
            .create(SHORT_OPT));
    }

    @Override
    public List<String> getUsageLines() {
        return Collections.singletonList(
            "[--" + LONG_OPT + "]");
    }

    @Override
    public boolean parseCommandLine(CommandLine commandLine) {
        skipArchived = commandLine.hasOption(SHORT_OPT);
        return true;
    }

    @Override
    public List<Version> selectVersions(List<Version> versions) {
        Stream<Version> stream = versions.stream();
        if (skipArchived) {
            stream = stream.filter(v -> CapStructHelper.getInt(v, ARCHIVED_PROPERTY)
                != 1);
        }
    }
}

```

```

    }
    return stream.collect(Collectors.toList());
  }
}

```

Example 3.20. Example for custom selector with parameter

NOTE

Note, that you must not add required parameters for version selectors within `fillInOptions` as the parameters will be marked as required even if other version selectors are used.



Content-UUID-Migration

This section describes three tools, which are meant for upgrade scenarios of *Content Management Server* in multi-environment setup, as for example a development and a production stage, related to UUIDs of contents. For details on UUID support in *CoreMedia Content Cloud* see Section 5.2, “UUIDs” in *Unified API Developer Manual*.

To finish the process, a server downtime is required. Thus, please plan your upgrade scenario carefully. Find more details at [section “Preparation” \[148\]](#).

Purpose

The goal of these tools is sharing the same UUIDs for contents between two servers in different environments.

To keep the UUID synchronized after this migration step, consider using `serverexport` and `serverimport` with enabled UUID export. For details see [Section “Serverimport/Server-export” \[180\]](#).

While being dedicated to this upgrade scenario, you may use the tools to synchronize UUIDs if you have forgotten to include the UUIDs during transfer with `serverexport` and `serverimport`.

Example Scenario

For the following descriptions the following example setup is expected:

Development Environment

Instance of *Content Management Server* used by developers.

Production Environment

Instance of *Content Management Server* used by editors. The assumption is, that this system receives contents from the development environment sometimes.

Typical Usage

The following is a typical flow of actions when migrating UUIDs. Assuming that your *Development Environment* is hosted at `dev.host` and your *Production Environment* is hosted at `prod.host` the following shows the order of commands. Remember, that you need to shut down your server at *Production Environment* when running the last command:

```
$ cm content-uuid-export \
  --user admin \
  --url https://dev.host/coremedia/ior \
  --query "BELOW PATH '/Sites'" \
  --dburl jdbc:mysql://dev.host:3306/cm_management \
  --dbuser cm_management \
  --dbpassword ... \
  --output dev-export.csv

$ cm generate-content-uuid-map \
  --dburl jdbc:mysql://prod.host:3306/cm_management \
  ...
  --input dev-export.csv \
  --output prod-map.csv

$ cm content-uuid-import \
  --dburl jdbc:mysql://prod.host:3306/cm_management \
  ...
  --input prod-map.csv
```

Example 3.21. Typical Usage Example

Note, that you not only have to shut down your server, but also restart any client. This applies to clients with the ability to reconnect as well, as they will not get notified on changed UUIDs.

Preparation

To finish the process of UUID synchronization you will require a server downtime for *Production Environment* when using the last of the three tools. For a rough estimation on possible downtimes see [section “Reference times for downtime during import” \[150\]](#). With a careful preparation you may reduce the impact of server downtime by incremental migration. For details see [section “Incremental Migration” \[149\]](#).



Live Servers not affected

Because live servers (master live server and replication live server) are unaware of content UUIDs, they require neither a shut down nor a restart. Same applies of course to all clients connected to live servers.

For details on support for UUIDs for contents see Section 5.2, “UUIDs” in *Unified API Developer Manual*.

The first two tools **content-uuid-export** and **generate-content-uuid-map** do not require a downtime. Nevertheless, it is recommended to execute these tools during limited editorial activity. This will reduce a possible performance impact of **content-uuid-export** and the generated data may be more consistent. Otherwise, data may contain for example undesirable references to contents deleted concurrently.

Unlike **content-uuid-export**, you may execute **generate-content-uuid-map** without running *Content Management Server*. This may be convenient, if you want to run **generate-content-uuid-map** and **content-uuid-import** on *Production Environment* without interruption.

The last of these tools, **content-uuid-import**, requires a downtime of the *Content Management Server* in the *Production Environment*. If you have any clients running with automatic reconnect, these need to be shut down as well, as the change to UUIDs will not be propagated to them until a restart.

All three tools require direct access to the database. The tools use the configuration available in `properties/corem/sql.properties`. If these properties do not match the database to access, you may specify the database connection options via command line parameters like `--dburl`. Thus, if applicable, ensure that you have the following properties at hand:

- `sql.store.url`,
- `sql.store.driver` [guessed from URL if not given],
- `sql.store.user`,
- `sql.store.password`, and possibly
- `sql.store.login-user-name` [required for some databases]

See Section 4.2.4, “Properties for the Connection to the Database” in *Deployment Manual* for details on these properties.

Incremental Migration

An incremental migration approach increases the effort, while decreasing the downtime of the servers and clients for one iteration. Having several options working with incre-

ments, choose the best of the options described in the following sections, or combine them. Whatever suits you best.

Incremental Export and Mapping

While **content-uuid-export** retrieves the content data by direct database access (due to performance reasons), you select the content to export by Unified API queries. For syntax and details see Section 5.7, “Query Service” in *Unified API Developer Manual*.

Split by Content Queries: It is recommended to split your migration tasks by paths. You may for example start with the root folder of your master site. Or you may start with everything outside of your sites folder. Using content queries provides a rich mechanism of partitioning your contents. For some examples on queries, consider calling **content-uuid-export** with `--query-help`.

After you exported your content, you can run **generate-content-uuid-map** with the results from **content-uuid-export**.

Incremental Import

The **content-uuid-import** provides the options `--from` and `--to`. These limit the processed elements provided by **generate-content-uuid-map**. `--from` specifies the first element to import. `--to` specifies the last element to import. If you run **content-uuid-import** with `--from 10` and `--to 20`, elements 10 to 20 (inclusive) will be processed.

How can I ensure that all UUIDs have been imported correctly? The key to this is, that **generate-content-uuid-map** will only output those mappings, which require the UUID of a content to be changed. Thus, it ignores especially UUIDs, which already match at *Production Environment* (for example from previous migration runs). Therefore, you could run **generate-content-uuid-map** again after your import. If this results in an empty mapping/file, all your UUIDs were mapped.

How to recover from aborted import? **content-uuid-import** can safely be aborted at any time. This, of course, also applies to unexpected abortions due to connection loss to database for example. To recover, you have several options, where the easiest one is just starting **content-uuid-import** again. This is, because updating the UUID twice (to the same value) does no harm to the system. To skip already processed elements, you may either run **generate-content-uuid-map** again prior to starting **content-uuid-import**, or you may adjust the parameters `--from` and `--to` accordingly.

Reference times for downtime during import

As stated before, while executing **content-uuid-import** the *Content Management Server* and related clients have to be stopped. This section will give you a rough overview on possible downtimes to expect for the last step.

Test Setup: A million contents were updated during the test. All databases were set up locally within Docker containers, each having memory of 5 GB and 8 CPUs. `content-uuid-import` was executed with `--threads 8` and `--batchsize 1000` (default).

Database	Duration in [s]
PostgreSQL 9.6	145
MySQL Database 5.7	728
DB2 10.5	24
MS SQL Server 2017-latest	353
Oracle 19.3.0-se2	32

Table 3.25. Reference times for `content-uuid-import` for a million contents

Command Reference

In the following you will get detailed help on the specific commands in the order of their execution during the migration process.

content-uuid-export

`content-uuid-import` is the first tool to be executed. It will generate a CSV file (more specifically a semicolon separated file, so that you may open it for example in several office applications directly). The CSV file will contain the following information, which are required to continue with the next tasks:

- Path** The path of the content.
- Type** The content type of the content.
- UUID** The UUID of the content.

Both, path and type, will be used to identify the content in *Production Environment*.

```
cm content-uuid-export [1] connection options
                        [2] database options
                        [3] query options
                        [ -? | --help ]
                        [{ -bs | --batchsize } count ]
                        [{ -E | --encoding } encoding ]
                        [ -M | --meta ]
                        [{ -o | --output } file ]
```

```
[ -v | --verbose ]

[1] { -u | --user } user [ { -d | --domain } domain ] [ { -p | --password } password ] [ --url IOR URL ]

[2] [ --dbdriver class ] [ --dbloginname name ] [ --dbpassword password ] [ --dburl JDBC URL ] [ --dbuser name ]

[3] { -q | --query | -qf | --queryfile } query or file [ { -l | --limit } limit ] [ { -e | --versions } ] [ { -qh | --query-help } ]
```

Example 3.22. Usage of *content-uuid-export*

The options have the following meaning:

Parameter	Description
<code>-? --help</code>	Will output usage information.
<code>{ -bs --batchsize } count</code>	The batch size defines how many elements are processed in one SQL statement. A larger number decreases the amount of queries against the database, and therefore increases the performance. However, if too high the SQL statement could become too large for the database, resulting in an I/O Error while connecting to the database. The default value is 1,000.
<code>--dbdriver class</code>	The database driver class. Overrides <code>sql.store.driver</code> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
<code>--dbloginname name</code>	The login username (needed for PostgreSQL on Azure for example). Overrides <code>sql.store.login-user-name</code> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
<code>--dbpassword password</code>	The database password. Overrides <code>sql.store.password</code> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
<code>--dburl JDBC URL</code>	The JDBC URL for the database. Overrides <code>sql.store.url</code> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
<code>--dbuser user</code>	The database user name. Overrides <code>sql.store.user</code> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
<code>{ -E --encoding } encoding</code>	Defines the encoding for the output. Default is UTF-8.

Parameter	Description
<code>{-l --limit} limit</code>	Query parameter, which limits the number of returned contents.
<code>-M --meta</code>	Will add meta information as line comments to the output. Meta information for example include the IOR URL, start and end time as well as contents processed. Note, that using this option typically prohibits opening the CSV file in third-party applications, as line comments are not part of the CSV standard.
<code>{-o --output} file</code>	Will output the CSV to the given file. The file will be created relative to your current working directory. If not set, output will be printed to console instead.
<code>{-q --query} query</code>	The query to select contents to export. Either this option or <code>--queryfile</code> must be given. For details on version see Section 5.7, "Query Service" in <i>Unified API Developer Manual</i>.
<code>{-qf --queryfile} file</code>	A file containing the query to execute. Convenient, if your operating system limits the character length for your command. Either this option or <code>--query</code> must be given. For details on version see Section 5.7, "Query Service" in <i>Unified API Developer Manual</i>.
<code>-qh --query-help</code>	Provides some examples for Unified API queries. If given, the tool will quit as soon as examples have been shown.
<code>-v --verbose</code>	Toggle verbose output.

Table 3.26. Parameters of *content-uuid-export*

generate-content-uuid-map

generate-content-uuid-map is the second tool to be executed. It will generate a CSV file (more specifically a semicolon separated file, so that you may open it for example in several office applications directly). The CSV file will contain the following information, which are required to continue with the next task:

- Content ID** The numeric ID of the content, which should be set to the given UUID.
- UUID** The UUID to be set for content identified by the content ID.

The ID will be used to identify the content in the database in *Production Environment* during the next task.

```
cm generate-content-uuid-map [(1) database options]
                             [-? | --help]
                             [{ -bs | --batchsize } count]
                             [{ -E | --encoding } encoding]
                             [{ -i | --input } file]
                             [-M | --meta]
                             [{ -o | --output } file]
                             [-v | --verbose]

(1) [--dbdriver class] [--dbloginname name] [--dbpassword
password] [--dburl JDBC URL] [--dbuser name]
```

Example 3.23. Usage of *generate-content-uuid-map*

Parameter	Description
-? --help	Will output usage information.
{ -bs --batchsize } count	The batch size defines how many elements are processed in one SQL statement. A larger number decreases the amount of queries against the database, and therefore increases the performance. However, if too high the SQL statement could become too large for the database, resulting in an I/O Error while connecting to the database. The default value is 1,000.
--dbdriver class	The database driver class. Overrides <i>sql.store.driver</i> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
--dbloginname name	The login username (needed for PostgreSQL on Azure for example). Overrides <i>sql.store.login-user-name</i> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
--dbpassword password	The database password. Overrides <i>sql.store.password</i> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
--dburl JDBC URL	The JDBC URL for the database. Overrides <i>sql.store.url</i> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .
--dbuser user	The database user name. Overrides <i>sql.store.user</i> . See Section 4.2.4, "Properties for the Connection to the Database" in <i>Deployment Manual</i> .

Parameter	Description
<code>{ -E --encoding } encoding</code>	Defines the encoding for input and output. Default is UTF-8.
<code>{ -i --input } file</code>	Input file generated by content-uuid-export to be processed. The file is relative to your current working directory. If unset, defaults to standard input instead.
<code>-M --meta</code>	Will add meta information as line comments to the output. Meta information for example include the IOR URL, start and end time as well as contents processed. Note, that using this option typically prohibits opening the CSV file in third-party applications, as line comments are not part of the CSV standard.
<code>{ -o --output } file</code>	Will output the CSV to the given file. The file will be created relative to your current working directory. If not set, output will be printed to console instead.
<code>-v --verbose</code>	Toggle verbose output.

Table 3.27. Parameters of *generate-content-uuid-map*

content-uuid-import

content-uuid-import is the third and last tool to be executed. It will update the contents identified by **generate-content-uuid-map** having the corresponding new UUID.

Prior to starting this tool, you need to shut down the *Content Management Server*. To reduce downtimes, you may want to read [section “Incremental Migration” \[149\]](#).

```
cm content-uuid-import [1] database options
    { -? | --help }
    [{ -bs | --batchsize } count ]
    [{ -E | --encoding } encoding ]
    [{ -F | --from } count ]
    [{ -i | --input } file ]
    [{ -T | --to } count ]
    [{ -t | --threads } threads ]
    [ -v | --verbose ]

[1] [--dbdriver class] [--dbloginname name] [--dbpassword password]
    [--dburl JDBC URL] [--dbuser name]
```

Example 3.24. Usage of *content-uuid-import*

Parameter	Description
<code>-? --help</code>	Will output usage information.
<code>{ -bs --batchsize } count</code>	The batch size defines how many elements are processed in one SQL statement. A larger number decreases the amount of queries against the database, and therefore increases the performance. However, if too high the sql statement could become to large for the database, resulting in an I/O Error while connecting to the database. The default value is 1,000.
<code>--dbdriver class</code>	The database driver class. Overrides <code>sql.store.driver</code> . See Section 4.2.4, “Properties for the Connection to the Database” in <i>Deployment Manual</i> .
<code>--dbloginname name</code>	The login username (needed for PostgreSQL on Azure for example). Overrides <code>sql.store.login-user-name</code> . See Section 4.2.4, “Properties for the Connection to the Database” in <i>Deployment Manual</i> .
<code>--dbpassword password</code>	The database password. Overrides <code>sql.store.password</code> . See Section 4.2.4, “Properties for the Connection to the Database” in <i>Deployment Manual</i> .
<code>--dburl JDBC URL</code>	The JDBC URL for the database. Overrides <code>sql.store.url</code> . See Section 4.2.4, “Properties for the Connection to the Database” in <i>Deployment Manual</i> .
<code>--dbuser user</code>	The database user name. Overrides <code>sql.store.user</code> . See Section 4.2.4, “Properties for the Connection to the Database” in <i>Deployment Manual</i> .
<code>{ -E --encoding } encoding</code>	Defines the encoding for input. Default is UTF-8.
<code>{ -F --from } count</code>	Position from which entry (inclusive, starting with 1) entries should be processed. Combine with <code>--to</code> to import UUIDs partially.
<code>{ -i --input } file</code>	Input file generated by generate-content-uuid-map to be processed. The file is relative to your current working directory. If unset, defaults to standard input instead.
<code>{ -t --threads } threads</code>	In order to increase performance, this program runs multi threaded. Therefore, this parameter defines how many threads are used to process the database queries. The default value is 4. If your system has more threads available it is recommended to set this parameter higher, as this will increase

Parameter	Description
	performance. The number of threads used also results to the amount of database connections being opened at the same time.
<code>{ -T --to } count</code>	Position up to which entry (inclusive, starting with 1) entries should be processed. Combine with <code>--from</code> to import UUIDs partially.
<code>-v --verbose</code>	Toggle verbose output.

Table 3.28. Parameters of *content-uuid-import*

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.

NOTE

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

The options have the following meaning:

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.

Table 3.29. Options of dbindex

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 [see [Section “Restoreusers” \[175\]](#)]. 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.

```
<group id="g13" name="global-manager" contentgroup="true" livegroup="false"
      administrative="false">
  <rule content="/Settings/Meta/Mail" type="CMMail" rights="RMDAP"/>
```

```
<rule content="/Settings/Options/Bundles" type="CMSettings" rights="RMDAP"/>
<rule content="/Settings/Options/Settings" type="CMSettings" rights="RMAP"/>
<rule content="/Settings/Taxonomies" type="CMTaxonomy" rights="RMDAP"/>
<rule content="/Themes" type="CMObject" rights="RMDAP"/>
<rule content="/Themes" type="CMTemplateSet" rights="RMDAP"/>
<rule content="/Themes" type="Folder_" rights="RMDAP"/>
<members>
  <group name="global-site-manager-c" contentgroup="true" livegroup="false"
    administrative="false">
    <rule content="/Sites/Chef Corp." type="CMObject" rights="RMDAPS"/>
    <rule content="/Sites/Chef Corp." type="Folder_" rights="RMDAPS"/>
    <rule content="/Settings/Taxonomies" type="CMTaxonomy" rights="RMDAP"/>

    <members>
      <user id="u10" name="Colin" home="/Home/Colin"/>
      <user id="u9" name="Rick C" home="/Home/Rick C"/>
    </members>
  </group>
</members>
</group>
```

Example 3.25. Snippet of dumpusers output

For a description of the flags shown at the `rights` attribute, see [Section 3.15.2, “User Rights Management” \[217\]](#).

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

```
cm dumpusers [[1] connection options]
              [{ -f | --file } file | [{ -e | --encoding } encoding
              ]]
              [ -ie | --include-uuids ]
              [ -P | --pretty ]
              [ -v | --verbose ]

[1] { -u | --user } user [{ -d | --domain } domain ] [{ -p | --password
} password] [-url IOR URL]
```

Example 3.26. Usage of dumpusers

The options have the following meaning:

Parameter	Description
<code>{ -e --encoding } <i>encoding</i></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 --file } <i>file</i></code>	File where to write dump users to. May be provided as URL. If unset, the XML will be printed to standard output.

Parameter	Description
<code>-ie --include-uuids</code>	Toggles export of UUIDs of users and groups. If contained, UUIDs will be automatically taken into account when restoring users, unless skipped explicitly. For Built-In Members Only While UUIDs will be written for built-in members as well as external members (for example provided by LDAP), the UUIDs for external members will be ignored when restoring users. It is expected, that your external user provider generates a stable unique ID. If for any reason the UUIDs for external members do not match, you will get an information about this issue.
<code>-P --pretty</code>	Pretty print file.
<code>-v --verbose</code>	Toggle verbose output.

Table 3.30. Parameters of dumpusers

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` 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` 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 `$INSTALL_DIR/etc/keys/<database name>.<dbuser>.rijndael`. Do not delete this key file and instead make sure that a backup exists in a safe place. Without the file, it is no longer possible to log in. You must copy this file to the Content Server installation under `$INSTALL_DIR/etc/keys` (The path can be configured by setting the property `cap.server.encrypt-passwords-key-file`). 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
```

JShell

The tool `cm jshell` is a convenience wrapper around Java's JShell. In short, it adds the CoreMedia classpath to JShell and provides a namespace for accessing a CoreMedia system via Unified API. To get to know details on JShell please consult the corresponding documentation, as for example the [Java Shell User's Guide \[Java 11\]](#). Find the UAPI Javadoc at <https://documentation.coremedia.com/cmcc-10> following the CMS Javadoc link.

As `cm jshell` just wraps the standard JShell, all command line options are the same as for the original JShell. Thus, you may get an overview of all options calling `cm jshell --help`.

Java Development Kit 9+ required

JShell is available since Java version 9. In addition to that, JShell is shipped with the Java Development Kit (JDK) only. It is not shipped with the Java Runtime Environment (JRE).

A corresponding failure is shown when starting `cm jshell` when JShell is not available, mentioning the requirements.



CoreMedia Namespace

`cm jshell` comes with a startup script, which behaves similar to JShell's built-in scripts `DEFAULT`, `JAVASE` and `PRINTING`. The startup script is named `COREMEDIA`.

Different to JShell's built-in scripts, you cannot reference `COREMEDIA` within JShell and it must be added via `--startup` command-line option.

Imports: First of all, `COREMEDIA` provides a bunch of imports, like for example `com.coremedia.cap.common.*` and `com.coremedia.cap.content.*`. For a list of all provided imports, call `/imports` within JShell.

Namespace: `COREMEDIA` defines a namespace (represented as variable), which provides some convenience for connecting to a CoreMedia system. There are two variables, which represent the very same namespace: `coremedia`, and a short version called `cm`.

Help: To get help for how to use this namespace, just call `coremedia.help()`. It will provide an overview of topics, which will guide you through the namespace. Along with that, you will find some examples in [section "Examples" \[165\]](#).

Initial Connection Setup

You may start up JShell in a way, that it automatically initializes a connection on startup phase. To do so, you have to provide the connection settings. You may choose one of the following configuration options, or even mix them:

1. [section "System Properties" \[163\]](#)
2. [section "Environment Variables" \[164\]](#)
3. [section "User Properties" \[164\]](#)

The given order applies to the precedence of properties. Thus, an IOR URL given as system property overrides the IOR URL from environment, whereas a connection password given as environment property overrides the password specified in the properties file in your home folder.

Unencrypted Passwords

The passwords provided directly via system properties, environment variables, or user properties are unencrypted. Take care to handle these settings with care, like protecting access to your home folder or by preventing calls to `cm jshell` to appear in command history.

In alternative to this, you may enforce a password prompt. For details see [section “Password Prompt” \[165\]](#).



System Properties

The `coremedia` namespace supports some system properties. To pass them to the JShell environment, you must specify them as remote options, using `-R<flag>`.

Supported system properties are:

System Properties

<code>repository.url</code>	The IOR URL of a Content Server to connect to. If specified, triggers an initial connection to the Content Server, which is available in the <code>coremedia</code> namespace via <code>coremedia.connection</code> . Requires credentials to be set along with the IOR URL.
<code>repository.user</code>	The user name to use when connecting to <code>repository.url</code> .
<code>repository.domain</code>	The user's domain to use when connecting to <code>repository.url</code> . Skip (or set to empty) to use no domain.
<code>repository.password</code>	The user's password to use when connecting to <code>repository.url</code> . To prevent exposing the password, you may use the password prompt approach as described in section “Password Prompt” [165] .

Beware of Untrusted Scripts

You should not pass passwords to untrusted scripts. The system property `repository.password` will expose the plain text password within JShell for interactive use, as well as to scripts.



User Properties

All properties described in [section “System Properties” \[163\]](#) may also be given in a properties file located in your home folder.

- `.cm.jshell.properties`
- `_cm.jshell.properties`, especially meant for older releases of Windows, where you may not create files starting with a dot.

Precedence: Compared to system properties and environment variables, properties in your home folder have the lowest precedence.

Environment Variables

Similar to system properties, you may specify the properties as environment variable. Note, that system properties take precedence. Thus, using system properties you may override any of the following environment variables.

Environment Variables

REPOSITORY_URL	The IOR URL of a Content Server to connect to. If specified, triggers an initial connection to the Content Server, which is available in <code>coremedia</code> namespace via <code>coremedia.connection</code> . Requires credentials to be set along with the IOR URL.
REPOSITORY_USER	The user name to use when connecting to <code>REPOSITORY_URL</code> .
REPOSITORY_DOMAIN	The user's domain to use when connecting to <code>REPOSITORY_URL</code> . Skip (or set to empty) to use no domain.
REPOSITORY_PASSWORD	The user's password to use when connecting to <code>REPOSITORY_URL</code> . To prevent exposing the password, you may use the password prompt approach as described in section “Password Prompt” [165] .

No Access for User-Provided Scripts

`cm jshell` passes environment variables to the remote script via an encrypted file. The data are exposed to the `coremedia` namespace only. User-provided remote scripts cannot access these environment variables.



Beware of Untrusted Scripts

You should not pass passwords to untrusted scripts. While the password passed via environment variables is not directly accessible, the system properties will expose details, which attackers may use to decode the password.



Password Prompt

In untrusted environments, you should not expose your password, neither in system properties provided via command-line, nor in environment, nor in user properties. Instead, you may configure `cm jshell` to prompt for a password.

To do so, you may either set a runtime system property, or you may set an environment property:

```
repository.pass
word.query
```

To use the runtime system property approach, add `-J-D repository.pass word.query=true` to your `cm jshell` command line options.

```
REPOSITORY_PASS
WORD_QUERY
```

As alternative to the runtime system property you may set environment variable `REPOSITORY_PASSWORD_QUERY` to `true`.

If you set any of the two to `true`, you will be prompted for a password.

Precedence: The password provided this way has a slightly higher precedence than a password provided via environment properties. In other words:

- It overrides a password given in user properties file.
- It overrides a password given as environment property.
- It is overridden by a password provided by remote system properties.

Examples

Below you will find some examples in addition to the internal help available via `core media.help()` within JShell. In Shell examples, commands are prefixed with `$`, while in JShell examples, commands are prefixed with `jshell1>`.

```
$ cm jshell
```

Example 3.27. Start JShell (Default)

Example 3.27, "Start JShell (Default) " [165]: Starts the JShell and automatically runs the startup scripts `DEFAULT` (from JShell) and `COREMEDIA`. The latter one provides CoreMedia imports as well as the `coremedia` namespace.

```
$ cm jshell --startup COREMEDIA
```

Example 3.28. Virtual Built-In Startup Script: COREMEDIA

Example 3.28, "Virtual Built-In Startup Script: COREMEDIA " [166]: Starts the JShell and runs startup scripts `COREMEDIA` only.

```
$ cm jshell --startup DEFAULT
```

Example 3.29. Restrict to DEFAULT

Example 3.29, "Restrict to DEFAULT " [166]: If you specify startup scripts explicitly not adding `COREMEDIA`, neither CoreMedia imports will be provided nor the `coremedia` namespace. This is very similar to starting the default JShell directly, with exception to the classpath, which includes the CoreMedia classpath.

```
$ cm jshell \  
-R"-Drepository.url=http://localhost:40180/ior" \  
-R"-Drepository.user=admin" \  
-R"-Drepository.password=admin"
```

Example 3.30. Connection at Startup (System Properties)

Example 3.30, "Connection at Startup (System Properties) " [166]: Starts the JShell and directly sets up a connection to the Content Server given by system properties. The connection will be available via `coremedia.connection`.

```
$ export REPOSITORY_URL=http://localhost:40180/ior  
$ export REPOSITORY_USER=admin  
$ export REPOSITORY_PASSWORD=admin  
$ cm jshell
```

Example 3.31. Connection at Startup (Environment Variables)

Example 3.31, "Connection at Startup (Environment Variables) " [166]: Starts the JShell and directly sets up a connection to the Content Server given by environment variables. The connection will be available via `coremedia.connection`.

```
$ cm jshell  
  
jshell> coremedia.connect(  
  "http://localhost:40180/ior",  
  "admin",  
  "admin")
```

```
jshell> coremedia.connection.getContentRepository()
```

Example 3.32. Connection within JShell

Example 3.32, "Connection within JShell" [166]: If you skipped specifying a connection to connect to at startup, you may always initiate a connection using the method `coremedia.connect`. Just as for the initial connection, the connection is available via `coremedia.connection` afterwards.

Note, that `coremedia.connection` and all related fields like `coremedia.cr` [ContentRepository] will always only refer to the latest established connection. You may open several connections in parallel, but you need to store them in your own variables.

To close the latest established connection, just call `coremedia.close()`.

```
$ cm jshell
jshell> /exit 42
```

Example 3.33. Exit Code Handling

Example 3.33, "Exit Code Handling" [167]: Within JShell or within your JShell script, you may exit JShell using the `/exit` command. You may specify an exit code which will then be the exit code of `cm jshell`. The default exit code is 0 [zero].

```
$ cm jshell
jshell> /save -start mystartup.jsh
jshell> /exit
$ cm jshell --no-startup ./mystartup.jsh
```

Example 3.34. Save Startup Script

Example 3.34, "Save Startup Script" [167]: If you want to customize the startup procedure, you may save the startup script using the `/save` command. In the given example the file will be saved to your current working directory. You can start JShell with your customized startup script afterwards. Specifying `--no-startup` ensures, that the default COREMEDIA startup script will not be triggered.

Groovy Shell



Deprecation Notice

As of *CoreMedia Content Cloud* 2004.1 `groovysh` is deprecated. Use `cm jshell` instead. For details see [Section "JShell" \[161\]](#).

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 `apps/content-server/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 `iorUrl` property to the corresponding URL of the *Content Server*. The profile defines some variables, which you can use to work with the repositories.

Configuration

Variable name	Calls	Description of return type
connection	<code>Cap.connect (params)</code>	<code>CapConnection</code> , the connection to the Content Server
cr	<code>connection.getContentRepository () ;</code>	<code>ContentRepository</code> , the content repository.
root	<code>cr.getRoot ()</code>	<code>Content</code> , the root folder of the content repository.
ac	<code>cr.getAccessControl ()</code>	<code>AccessControl</code> , all access control aspects of content.
pu	<code>cr.getPublicationService ()</code>	<code>PublicationService</code> , all publication aspects of content.
qs	<code>cr.getQueryService ()</code>	<code>QueryService</code> , query aspects of content. Allows making structured queries against the content repository.

Variable name	Calls	Description of return type
ur	<code>connection.getUserRepository()</code>	<code>UserRepository</code> , gives access to users and groups.
wr	<code>connection.getWorkflowRepository()</code>	<code>WorkflowRepository</code> , gives access to workflows.

Table 3.31. Variables for Groovy Shell

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.

Using the Groovy Console

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 <https://documentation.coremedia.com/cmcc-10> 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> 
```

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 of jmxdump

```
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
```

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. For example.: service:jmx:rmi:///<server URL>:<JMX server port>/jndi/rmi:///<server URL>:<JMX registry>/jmxrmi
-v	More verbosity

Table 3.32. Options of jmxdump

The default ports for the URL parameter can be found in Chapter 4, *CoreMedia Properties Overview* in *Deployment Manual*.

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” \[111\]](#)].

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
-u,--user <user name>        user for login (required)
```

Usage of killsession

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

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

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.

NOTE

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

You must set the `CM_KEYSTORE_LOCATION` and `CM_KEYSTORE_PASSWORD_FILE_LOCATION` system properties not only for the `encryptpasswordproperty` tool, but also for the tool that uses the encrypted password, for instance `schemaaccess`.

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 3.35. Usage of the events utility

The events utility has the following options:

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 <code>com.coremedia.cap.content.Timestamp#fromNumbers(int,int,int)</code> for further details. Note that session events don't provide timestamps.

Table 3.33. Options of the events utility

```
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)
```

Example 3.36. Output of events

Restoreusers

The `restoreusers` tool reads an XML file which has been written using the `dumpusers` tool [see [Section “Dumpusers” \[158\]](#)]. 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. 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.

```
<group id="g13" name="global-manager" contentgroup="true" livegroup="false"
  administrative="false">
  <rule content="/Settings/Meta/Mail" type="CMMail" rights="RMDAP"/>
  <rule content="/Settings/Options/Bundles" type="CMSSettings" rights="RMDAP"/>

  <rule content="/Settings/Options/Settings" type="CMSSettings" rights="RMAP"/>

  <rule content="/Settings/Taxonomies" type="CMTaxonomy" rights="RMDAP"/>
  <rule content="/Themes" type="CMObject" rights="RMDAP"/>
  <rule content="/Themes" type="CMTemplateSet" rights="RMDAP"/>
  <rule content="/Themes" type="Folder_" rights="RMDAP"/>
  <members>
    <group name="global-site-manager-c" contentgroup="true" livegroup="false"
      administrative="false">
      <rule content="/Sites/Chef Corp." type="CMObject" rights="RMDAPS"/>
      <rule content="/Sites/Chef Corp." type="Folder_" rights="RMDAPS"/>
      <rule content="/Settings/Taxonomies" type="CMTaxonomy" rights="RMDAP"/>

      <members>
        <user id="u10" name="Colin" home="/Home/Colin"/>
        <user id="u9" name="Rick C" home="/Home/Rick C"/>
      </members>
    </group>
  </members>
</group>
```

Example 3.37. Snippet of dumpusers output

For a description of the flags shown at the `rights` attribute, see [Section 3.15.2, “User Rights Management” \[217\]](#).

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.



Restrictions

Many attributes cannot be changed once a group or user has been created:

- member names, as members are identified by name;
- domain;
- UUID, unless `--force-uuids` is specified;
- flags `isContentGroup` and `isLiveGroup`;

and in addition to that for externally provided users (like from LDAP):

- password;
- UUID;
- members of groups.

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.

```
cm restoreusers [[1] connection options]
    { { -f | --file } file | { -z | --zip } file | { -zd |
    --zip-directory } directory } }
    [ --skip-uuids | --force-uuids ]
    [ -v | --verbose ]

[1] { -u | --user } user [ { -d | --domain } domain ] [ { -p | --password
} password ] [ -url IOR URL ]
```

Example 3.38. Usage of `dumpusers`

The options have the following meaning:

Parameter	Description
<code>{ -f --file } file</code>	File where to read the user repository XML from. May be provided as URL.
<code>--force-uuids</code>	Enforces setting UUIDs for existing built-in members. While, by default, existing built-in members will not get a new UUID assigned, you may enforce overriding UUIDs with this option. This option has no effect, if <code>--skip-uuids</code> is given.

Parameter	Description
	Override UUIDs With Care If you used member UUIDs in references, your references will become invalid. Mapped UUIDs will be written to output, so that you may adapt the references.
<code>--skip-uuids</code>	Ignores UUID possibly given in user repository XML. Instead, when creating built-in members, random UUIDs will be generated. Implicit Skip on Server Version or Type If the server you are connected to does not support creating members with given UUID (because of type or version, see below), UUIDs will be implicitly ignored. Creating members with UUID is neither supported for versions prior to 2010.1 nor for Live Servers, that is, it is only supported for Content Management Servers 2010.1 and later.
<code>{ -z --zip } file</code>	Alternatively to the <code>--file</code> option, you can specify a zip file to extract user repository XML files from.
<code>{ -zd --zip-directory } directory</code>	A subdirectory within the zip file.
<code>-v --verbose</code>	Toggle verbose output.

Table 3.34. Parameters of `restoreusers`

Exit Code

On any failure while restoring users, `restoreusers` will exit with a code different to 0 (zero).

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 of runlevel

```
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
```

The options have the following meaning:

Parameter	Description
-g	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.
-w	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, 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.
-a	Abort a pending runlevel switch triggered by a recent invocation of <code>runlevel</code> .

Parameter	Description
<code>-r</code>	Specify the new runlevel. The legal values are described in the Operations Basics

Table 3.35. Options of runlevel

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.

WARNING

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



```
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
```

Example 3.39. Usage of `schemaaccess`

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 `properties/corem/sql.properties`.

The actions have the following meaning:

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.

Table 3.36. *Schemaaccess actions*

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 [source] do not have to match those on the import server [target]. 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 *Blueprint Developer Manual*.

UUIDs – Universally Unique Identifiers

`serverexport` and `serverimport` provide an option to transfer UUIDs from the export server [source] to the import server [target].

Limitations:

- **Documents Only:** Because `serverexport` and `serverimport` create folders only implicitly, that is, there is no artifact in `serverexport` explicitly referring to a folder, UUIDs for folders are not going to be transferred between two systems.
- **New Content Only:** Only newly created documents will get the UUID set. This is because you may have used the UUID of the target system already as reference.
- **Content UUIDs Only:** As `serverexport` and `serverimport` do not deal with versions, UUIDs of versions are not synchronized between source and target system.
- **Paths First – Issues with Duplicate UUIDs:** As `serverexport` and `serverimport` use the path information to identify identical contents, conflicts for UUIDs may be reported if you moved/renamed contents in source and target system. To resolve issues like this, you may want to use `--skip-uuids` option for `serverimport`.

Upgrade Scenarios:

If you upgraded from versions prior to 2010.1, or if you forgot to include UUIDs for such a transfer scenario once, you may want to read [Section “Content-UUID-Migration” \[147\]](#).

For details on UUID support in *CoreMedia Content Cloud* see Section 5.2, “UUIDs” in *Unified API 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>-z, --zip <URL></code>	Import the contents of the zip file. This is equivalent to manually extracting the zip and importing the extraction directory.
<code>-zd, --zip-directory <directory/within/zipfile></code>	As a refinement of the <code>-z</code> option, you can restrict the import to a particular directory within the zip file.
<code>-h, --halt</code>	Halt on error. Using this option is recommended to prevent any inconsistent state of the imported content, like for example links to embedded images in Richtext removed, because they are missing on server and in server export files.
<code>-v, --verbose</code>	enables verbose output

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>--skip-uuids</code>	If the export files contain UUIDs (see option <code>--include-uuids</code> for <code>serverexport</code>), you may skip importing them. This will generate a random UUID for each newly created content. This option may also be used to resolve problems on import of contents with UUID. You may struggle for example with duplicate UUIDs, if you moved your contents in source or target system. This option will ignore the problem and create a random UUID instead. To resolve partial duplicate UUIDs while creating contents with UUIDs where possible, first run without this option and repeat the very same import afterwards with this option enabled. For details see section "UUIDs – Universally Unique Identifiers" [181].
<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.

Parameter	Description
<code>-url <ior url></code>	The IOR URL of the Content Server.

Table 3.37. Parameters of the `serverimport` utility

On executing the program, a zip URL or 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> | -z <URL> ] [other options] [ <id> | <uuid> | <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.
<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>-z, --zip <URL></code>	Create a zip file of the exported contents. The URL must be writable. Currently, file URLs and s3 URLs (AWS) are known to work. s3 URLs assume that your environment provides suitable credentials. For details see Working with AWS Credentials .
<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.

Parameter	Description										
<code>-U, --include-uuids</code>	By default, UUIDs of contents are not exported. As a result, contents created on import will get random UUIDs. If you add this option, UUIDs will be added to the export files. UUIDs in export files will be automatically taken into account by <code>server import</code>, unless you use the option <code>--skip-uuids</code>. For details see section "UUIDs – Universally Unique Identifiers" [181].										
<code>-l, --lint <warning></code>	Enables different warnings to be logged. The following warnings can be specified: <table> <tr> <td><code>linkignored</code>(default)</td><td>Warn on links to either destroyed or deleted contents which are ignored on export.</td></tr> <tr> <td><code>linkmissing</code></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><code>translationstate</code></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><code>all</code></td><td>Warn on all detected issues.</td></tr> <tr> <td><code>none</code></td><td>Disable warnings.</td></tr> </table>	<code>linkignored</code> (default)	Warn on links to either destroyed or deleted contents which are ignored on export.	<code>linkmissing</code>	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.	<code>translationstate</code>	Warn on translation states (see Multi-Site) which are considered harmful especially because they cannot be rebuilt exactly on import.	<code>all</code>	Warn on all detected issues.	<code>none</code>	Disable warnings.
<code>linkignored</code> (default)	Warn on links to either destroyed or deleted contents which are ignored on export.										
<code>linkmissing</code>	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.										
<code>translationstate</code>	Warn on translation states (see Multi-Site) which are considered harmful especially because they cannot be rebuilt exactly on import.										
<code>all</code>	Warn on all detected issues.										
<code>none</code>	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.										
<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.										

Parameter	Description
<code>--blobsize-limit <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>sharedblob basedir</code> .
<code>-s, --sharedblob-basedir <sharedblob-basedir></code>	The directory in which to store blobs that are larger than the size given by <code>blobsize-limit</code> . Equal blobs will be stored only once.

Table 3.38. Parameters of the `serverexport` utility

As optional arguments, the IDs, UUIDs 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



WARNING

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 4.2.4, “Properties for the Connection to the Database” in *Deployment Manual*].

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

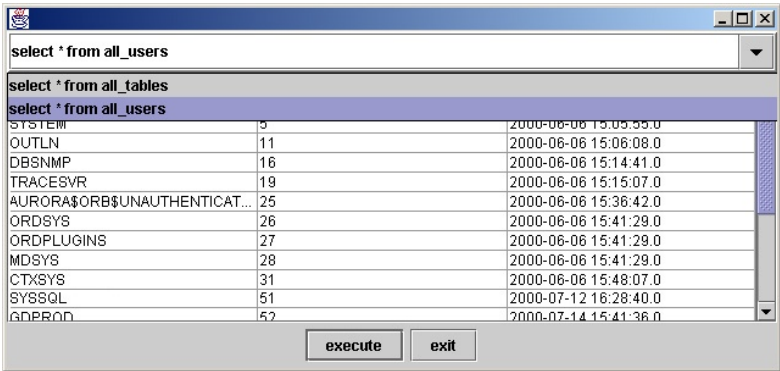


Figure 3.2. CM SQL user interface

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 3.40. CM sql command line operation

NOTE

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 Section 4.7, “Logging” in *Operations Basics* for details on the logging of *CoreMedia CMS*.

CAUTION

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” [111]).
-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 ses-

Parameter	Description
	sions, unknown denotes all sessions but those of the concrete type values.

Table 3.39. 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, 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/capserver-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
```

The tool needs a properly configured `sql.properties` file so that it runs against the database of the *Content Server*

Usedlicenses

CoreMedia CMS uses a license system based - among others - on named users (see the Operations Basics 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, feeder, editor, importer, publisher, replicator, studio, system, webserver, workflow.

You can find the ID of a user in the *User Manager* of *Studio* 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
-print	Prints out the named licenses in use
-deleteuser <user>	Deletes all licenses (thus making them available for reuse) for the user indicated by either the user id or the user name

Table 3.40. Parameters of the *usedlicenses* utility

Example

A new user tries to log in *Studio* 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.

3.13.3 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>... | -U <uuidl>... |
        -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)
-U,--uuid <uuid>            UUID(s) to select contents
-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:

Parameter	Description
-c	Ids of resources to be approved. Ids have the following format: coremeda:///cap/content/1234.
-v	Ids of content item versions to be approved. Ids have the following format: coremeda:///cap/version/1234/56. 56 must be replaced by the version number you want to approve.
-cq <query>	Content Selection Content Query to locate contents. Parameter can be used multiple times. Results will be linked by OR.
-I <id>	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 linked by OR.
-U <UUID>	Content Selection The UUID of a content to be approved. Parameter can be used multiple times. Results will be linked by OR.
-t <path>	Content Selection The path of a resource to be approved. Parameter can be used multiple times. Results will be linked by OR.
<ids>	Content Selection IDs to be approved. If mixing with other content selection parameters, it is recommended to explicitly use the parameter -I Parameter can be used multiple times. Results will be linked by OR.
-l <limit>	Content Selection Limit the number of selected contents. Default is unlimited.

Table 3.41. Options of approve

NOTE

For information about the UAPI query syntax see Section 5.7, “Query Service” in *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.

```
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 multi-master
                             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
```

Example 3.41. Usage of bulkpublish

The options have the following meaning:

Parameter	Description
-a	Equivalent to the combination of -ap and -av.
-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 /articles/sport). If no path is entered, all resources below the root folder will be used. If you use a multi-master enabled system, it is mandatory to enter a path to a folder that belongs to a single base folder (see Section 2.3, “Multi-Master Publishing” [23]). This option can be given multiple times.

Table 3.42. Options of the *bulkpublish* tool

Destroy

With the `destroy` tool you can delete resources from the repository.

WARNING

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: cm destroy -u <user> <other options> [ -f ] [ -R ]
       [ -l <limit> ] [ <idl>... | -I <idl>... |
       -U <uuidl>... | -t <pathl>... | -cq <queryl>... |
```

Usage of *destroy*

```

    -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)
-U,--uuid <uuid>          UUID(s) to select contents
-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:

Parameter	Description
-f	Force the destruction of non-empty folders, which is normally forbidden due to safety considerations
-cq <query>	Object Selection Content Query to locate contents. For versions see -vq. Parameter can be used multiple times. Results will be linked by OR.
-vq <query>	Object Selection Version Query to locate versions. For contents see -cq. Parameter can be used multiple times. Results will be linked by OR.
-R	Object Selection Add contents from recycle bin. Could also be done via -cq.
-I <id>	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 linked by OR.
-U <uuid>	Object Selection Select contents via UUID. This is an alternative if you want to mix UUIDs with for example content queries. Parameter can be used multiple times. Results will be linked by OR.
-t <path>	Object Selection Path of a resource to be destroyed. Parameter can be used multiple times. Results will be linked by OR.
<ids>	Object Selection IDs of objects to be destroyed. If mixing with other content selection parameters, it is recommended to explicitly use the parameter -I. Parameter can be used multiple times. Results will be linked by OR.

Parameter	Description
<code>-l <limit></code>	Object Selection Limit the number of selected objects to the given value. Default is unlimited.

Table 3.43. Options of destroy

NOTE

For information about the UAPI query syntax see Section 5.7, “Query Service” in *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.

WARNING

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” \[137\]](#)].

Concerning users and groups, `destroy` is not an LDAP tool. If you apply `destroy` to LDAP users, the users will only be logged out and synchronized with the LDAP repository on the next access. If you apply `destroy` to LDAP groups, the groups will lose their roles and be synchronized with the LDAP repository on the next access. Unless the users have been deleted from the LDAP server, they will not disappear from the *CoreMedia CMS*. 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 expedite the synchronization process and remove special privileges immediately.

Publish

With the `publish` tool you can publish resources.

Usage of publish

```
usage: cm publish -u <user> <other options> [ -l <limit> ]
      [ <idl>... | -I <idl>... | -U <uuid>... |
      -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)
-U,--uuid <uuid>              UUID(s) to select contents
-ur,--url <url>               url to connect to
-v,--verbose                  enables verbose output
```

The options have the following meaning:

Parameter	Description
<code>-cq <query></code>	Content Selection Content Query to locate contents. Parameter can be used multiple times. Results will be linked by OR.
<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 linked by OR.
<code>-U <UUID></code>	Content Selection The UUID of a content to be published. Parameter can be used multiple times. Results will be linked by OR.
<code>-t <path></code>	Content Selection The path of a resource to be published. Parameter can be used multiple times. Results will be linked by OR.
<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 linked by OR.
<code>-l <limit></code>	Content Selection Limit the number of selected contents. Default is unlimited.

Table 3.44. Options of publish



NOTE

For information about the UAPI query syntax see Section 5.7, “Query Service” in *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 3.42. 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).

Parameter	Description
<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 publishall 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 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>

Table 3.45. Parameters of the publishall utility

Because two passwords are used, it is not possible to provide credentials using the environment variable `REPOSITORY_PASSWORD`.

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-master 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” \[80\]](#).

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 `republish` 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 3.8.3, "Recovery of a Master Live Server Database" \[56\]](#)].

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" \[198\]](#) 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 for any reason, no content items linking to that content item can be published.

```
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 multi-master 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

Example 3.43.

The options have the following meaning:

Parameter	Description
<code>-a</code>	Start date for content items to be republished. The format is <code>yyyyMMddHHmmss</code> . By default, the start date is two days before now.
<code>-b</code>	The end date for content items to be republished. The format is <code>yyyyMMddHHmmss</code> . By default, the end date is the current date.
<code>-f</code>	The base folder for republication. It's only mandatory for multi-master systems. By default, the root folder will be taken as the base folder.

Table 3.46. Parameters of the republish utility

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.

NOTE

For more details about the query syntax and for more examples see Section 5.7, “Query Service” in *Unified API Developer Manual*.



```
usage: cm query -u <user> <other options> [ -xl <limit> ] [ -R ]
      [ -l <limit> ]
      [ <id1>... | -I <id1>... | -U <uuid1>... |
      -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
```

```
-p,--password <password>      into a UAPI query
-q,--query <query>             password for login
                                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)
-U,--uuid <uuid>              UUID(s) to select contents
-ur,--url <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 3.44. Query usage

Parameter	Description
-cq <query>	Content Selection Content Query to locate contents. For versions see -vq. Parameter can be used multiple times. Results will be linked by OR.
-vq <query>	Content Selection Version Query to locate contents. For contents see -cq. Parameter can be used multiple times. Results will be linked by OR.
-R	Content Selection Add contents from recycle bin. Could also be done via -cq.
-I <id>	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 linked by OR.
-q <query>	The query string. Especially to be used for transforming legacy queries at command line. For selection of contents the parameters -cq and -vq are recommended. Parameter can be used multiple times. Results will be linked by OR.
-e	If set, the query specified by -q searches for all versions of the content items. <i>Deprecated:</i> Use -vq instead.
-xl <limit>	Limit the number of results for executing transformed legacy queries. If not given defaults to -l.
-m	With the -m option you can transform legacy queries (according to the coremedia-query.dtd DTD) into UAPI queries (according to the

Parameter	Description
	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>-U <UUID></code>	Content Selection Select contents via UUID. This is an alternative if you want to mix UUIDs with for example content queries. Parameter can be used multiple times. Results will be linked by OR.
<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 linked by OR.
<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 linked by OR.
<code>-l <limit></code>	Content Selection Limit the number of selected contents to the given value. Default is unlimited.

Table 3.47. Parameters of the query utility

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 3.45. 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 3.46. 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 ::=
  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
;
```

Example 3.47. EBNF definition of the query language

The next table shows the definitions of the identifiers and literals of the query language.

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.

Item	Example	Description
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	'coremedia:///cap/content/1'	ID literals denote a content, version or user by giving its ID, as returned by CapObject.getId() .

Table 3.48. Identifiers and literals

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.

Queryapprove usage

```
usage: cm queryapprove -u <user> [other options]
      [-i] [-e] [-l <limit>] -q <query>

available options:
-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
```

<code>-v,--verbose</code>	enables verbose output
---------------------------	------------------------

Parameter	Description
<code>-e</code>	If set, the query searches for all versions of the content items.
<code>-i</code>	Include unapproved parent folders automatically into approval
<code>-l <limit></code>	Limit the number of results to the given value.
<code>-q <query></code>	The query string.
<code>-qf <queryfile></code>	A file that contains the query string.

Table 3.49. 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.

Querypublish usage

```
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
```

<code>-v,--verbose</code>	enables verbose output
---------------------------	------------------------

Parameter	Description
<code>-e</code>	If set, the query searches for all versions of the content items.
<code>-l <limit></code>	Limit the number of results to the given value.
<code>-q <query></code>	The query string.
<code>-qf <queryfile></code>	A file that contains the query string.

Table 3.50. Parameters of the `querypublish` utility

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.

```
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
```

Example 3.48. Usage of search utility

Parameter	Description
<code>-t</code>	Only search for content items of the given content type and inherited types.
<code>-q</code>	The string to search for.

Parameter	Description
<code>-f</code>	Only search in the given folder and its subfolders.

Table 3.51. Parameters of the search utility

3.14 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 5.2, "Managed Properties" \[280\]](#) in order to get an overview of the managed properties of the *Content Server*.

CAUTION

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.



3.15 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.

CAUTION

The following sections describe the technical basis of user management with *CoreMedia Content Cloud*. If you want to learn how to manage users with CoreMedia Studio, see Section 3.5, “Managing Users and Groups” in *Studio User Manual*.



3.15.1 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.

Groups	Description
system	Reserved for system users with read-only access, do not use for other users
system-write	Reserved for system users with full access, do not use for other users. Exists on the Content Management Server only.
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 3.52. 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	replicator studio watchdog webserver feeder

Table 3.53. Users and their groups

Content Management Server:

Groups	Users
system-write	publisher

Groups	Users
	workflow
importer	importer
approver	no default user
publisher	no default user
chief editor	no default user
editor	no default user

Table 3.54. Users and their groups in Content Management Server only

The default mapping of user rights to users, resources and resource types is as follows:									Default mapping of user rights
Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
ad-min-is-trat-ors	/	all	X	X	X	X	X	X	
	/	Folder	X			X	X	X	X
sys-tem	/	all	X	-	-	-	-	-	
	/	Folder	X			-	-	-	-
sys-tem-write	/	all	X	X	X	X	X	-	
	/	Folder	X			X	X	-	X

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
im-port-er	/	all	X	X	X	X	X	-	
	/	Folder	X			X	X	-	X
ap-prover	/	all	-	-	-	X	-	-	
	/	Folder	-			X	-	-	-
pub-lish-er	/	all	-	-	-	-	X	-	
	/	Folder	-			-	X	-	-
chief edit-or	/	all	X	X	X	X	X	-	
	/	Folder	X			X	X	-	X
edit-or	/	all	X	X	X	-	-	-	
	/	Folder	X			-	-	-	-
edit-or	/System	all	X	X	-	-	-	-	
	/System	Folder	X			-	-	-	-

Table 3.55. Default mapping of user rights

Workflow Role Groups

The CoreMedia Workflow installation comes with the following predefined workflows which cover the publication and translation of resources:

- simple-publication
- two-step-publication
- translation

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
translation-manager-role	A user of this group may start a translation workflow
administratoren	A user of this group has some rights on specific workflows for administrative tasks

Table 3.56. 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 Workflow Server properties.

```
# remap roles for default workflows
#workflow.map-role.approver-role=approver@example.org
#workflow.map-role.composer-role=composer@example.org
#workflow.map-role.publisher-role=publisher@example.org
#workflow.map-role.translation-manager-role=translation@example.org
```

```
#workflow.map-role.administratoren=administrators@example.org
```

Example 3.49. Groups in Workflow Server properties

These are looked up only once when uploading the workflow. See the Workflow Manual for a more detailed description of Workflow Server properties.

The translation-manager-role must match the setting in the property `sitemodel.translationManagerRole` of the site model, see Section 5.5, “Localized Content Management” in *Blueprint Developer Manual* for details.

Changing the Password of System Users

The CoreMedia system comes with predefined system users (see Table 3.53, “Users and their groups” [212]) that are used by other components to connect with the content server.

System user	Used by	Password property
feeder	Content Feeder CAE Feeder	repository.password
publisher	Publisher	<code>publisher.local.password</code> <code>publisher.target[0].password</code>
		The publisher connects with both, <i>Content Management Server</i> and <i>Master Live Server</i> and uses therefore two properties with passwords.
studio	Studio Server	repository.password
watchdog	Watchdog, used in watchdog definitions	password
webserver	CAE Elastic Worker	repository.password
workflow	Workflow Server	workflow.password

Table 3.57. System users and usage

If you want to change the password of a system user, you can either set new passwords before the first start of the Content Server using a property or change it later, using the user management.

Changing password at first start

You can use the property `cap.server.initialPassword.<username>` [see [Section 5.1, “Configuration Property Reference” \[278\]](#)] to change the default passwords before the first start of the Content Server.

1. Change the property for the users for whom you want to change the password in the Content Server, for example `cap.server.initialPassword.websrvr=yourpassword`.
2. Change the password in the property file of the components that connect with this user.

Change the password with user management

1. Change the password at the Content Server. Either use the user management in *Studio* [see [Section 3.5, “Managing Users and Groups” in *Studio User Manual*](#)] or use the `changepassword` server tool [see [Section “Changepassword” \[136\]](#)].
2. Change the password in the property file of the components that connect with this user.

If you think, that a plain text password in a properties file is a security threat, then you can use the `encryptpasswordproperty` tool [see [Section “Password Property Encryption” \[171\]](#)] to encrypt the passwords in the properties files.

3.15.2 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 '+':** Internally, the folder type is stored as the value "+".
- **User:** Users may operate on resources, if they have 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 and the possible resource operations:
- **Rights flag:** The rights flag is used, for example, in the *dumpusers* tool to show the applied rights in a short way.

You will edit rights in *Studio* [see Section 3.5, “Managing Users and Groups” in *Studio User Manual*]. There, the denomination of rights is slightly different from the UAPI names. [Table 3.58, “User rights” \[218\]](#) shows both versions.

Right	Site Manager notation	Rights Flag	Description
READ	READ	R	Read content names, content items content and folder names
WRITE	EDIT	M	Create, check out, check in, rename, move and save content items. In the Site Manager, you cannot directly attach the EDIT right to a folder. You have to use the FOLDER right.
DELETE	DELETE	D	Mark and unmark a content item for deletion, move an item to trash. In the Site Manager, you cannot directly attach the DELETE right to a folder. You have to use the FOLDER right.
APPROVE	APPROVE	A	Approve, disapprove, approve place, disapprove place a content item or folder
PUBLISH	PUBLISH	P	Publish a resource
	FOLDER		create subfolder, rename, move and delete a folder The FOLDER right only appears in the Site Manager's user management, but not in the Unified API or Studio. It is a visualization of the DELETE and WRITE rights, attached to a folder. Therefore, DELETE and WRITE rights of a folder must always have the same value.
SUPERVISE	SUPERVISE	S	Check in or uncheckout a content item from a different user, grant new rights

Table 3.58. 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 x RESOURCES x RESOURCE
TYPES x RIGHTS)
where
GROUPS is the set of groups
RESOURCES is the set of resources
RESOURCETYPES is the set of resource types and the 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

Group	Re- source	Re- source Type	READ	EDIT	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
	/F1	Folder	X			-	-	-	-

Table 3.59. Example rule table

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, 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	X	-	-	-	-	-	

Table 3.60. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-	X	-	-	-	-	

Table 3.61. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2/doc	Article	-	X	-	-	-	-	

Table 3.62. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-	X	-	-	-	-	
G	/F1/F3	Article	-	X	-	-	-	-	

Table 3.63. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1F2/doc	Article	-	-	X	-	-	-	

Table 3.64. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1F2/doc	Article	-	-	X	-	-	-	
	/F1/F2	Article		X					

Table 3.65. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1F2/doc	Article	-	-	-	X	-	-	

Table 3.66. Rule to (dis)approve a content item

Operations:

- publish content item *doc*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1F2/doc	Article	-	-	-	-	X	-	

Table 3.67. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1F2/doc	Article	-	-	-	-	-	X	

Table 3.68. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Folder	X			-	-	-	-

Table 3.69. Rule to read folder properties

Operations:

- place approve folder *F2*
- place disapprove folder *F2*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Folder	-			X	-	-	-

Table 3.70. Rule to place approve or disapprove a folder

Operations:

- publish folder *F2*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Folder	-			-	X	-	-

Table 3.71. Rule to publish a folder

Operations:

- create a subfolder in folder F2

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Folder	-			-	-	-	X

Table 3.72. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Folder	-			-	-	-	X

Table 3.73. Rule to operate on subfolders

Operations:

- move folder *F2* to folder *F3*

Required rights:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Folder	-			-	-	-	X
	/F3	Folder	-			-	-	-	X

Table 3.74. Rules to move a folder

NOTE

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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F/F2/doc	Article	-			-	-	X	-

Table 3.75. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	-			-	-	X	-

Table 3.76. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Folder	-			-	-	X	-

Table 3.77. 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	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
	/F1	Folder	X			-	-	-	-

Table 3.78. 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:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G1	/F1	Article	X	X	-	-	-	-	
	/F1	Folder	X			-	-	-	-
G2	/F1	Article	X	-	X	-	-	-	
G1	/F1/F2	Article	X	-	-	X	-	-	

Group	Re- source	Re- source Type	READ	EDIT	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
	/F1/F2	Folder	X			-	-	-	-
G1	/F1	ShortArticle	X	X	-	-	X	-	

Table 3.79. Example for conflicting rules

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:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	
	/F1	Folder	X			-	-	-	-
G	/F2	Article	X	-	-	X	-	-	
	/F2	Folder	X			-	-	-	-

Table 3.80. Example rules to compute effective rights

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:

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1/F2	Article	X	X	-	-	-	-	

Group	Re- source	Re- source Type	READ	EDIT	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
	/F1/F2	+	X			-	-	-	-

Table 3.81. Example rules with implicit navigate through right

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	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
G	/F1	+	X			-	-	-	-
G	/F1/F2	Article	X	X	-	-	-	-	
	/F1/F2	+	X			-	-	-	-

Table 3.82. 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	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
G	/F1	Article	-	X	-	-	-	-	
	/F1	+	X			-	-	-	-

Table 3.83. 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	DE- LETE	AP- PROVE	PUB- LISH	SUPER- VISE	FOLDER
G	/F1	Article	X	X	-	-	-	-	

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
	/F1	Folder	X			-	-	-	-

Table 3.84. Example rules with explicit READ right

3. 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.

Group	Re-source	Re-source Type	READ	EDIT	DE-LETE	AP-PROVE	PUB-LISH	SUPER-VISE	FOLDER
G	/F1	Folder	-			-	-	-	-
G	/F1/F2	Article	X	X	-	-	-	-	
	/F1/F2	Folder	X			-	-	-	-

Table 3.85. Example rules with READ right withdrawn

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 Studio in the User Manager in the *Effective Rules* tab.

3.15.3 Administrator Groups

Members of administrator groups are supposed to do administrative work in the Core-Media 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-master 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.

3.15.4 Content Server Groups and Users

CoreMedia CMS has some groups and users for standard operation (so called built-in users/groups see [Section 3.15.1, "Predefined Users and Groups" \[211\]](#), such as workflow) on the *Content Management Server*. You can add 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 *Studio* 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 3.12.3, "LdapUserProvider" \[94\]](#) 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

```
cap.server.userproviders[#].ldap.content-management-groups=true
cap.server.userproviders[#].ldap.live-groups=true
```

in the `contentserver` application properties.

If you want to connect to another LDAP server you can extend the `LdapUserProvider` class for your own user provider (see [Section 3.12.3, "LdapUserProvider" \[94\]](#) 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 Manager* of *Studio* to add rules to your groups. Read [Section 3.15.2, "User Rights Management" \[217\]](#) for details on rights and rules.

3.15.5 Live Server Groups and Users

CAUTION

CoreMedia Live Server Group management is not used anymore and has therefore been deprecated.

CoreMedia is planning to remove Live Server Group management together with Site Manager from the product portfolio with the next major release.



CoreMedia CMS needs some groups and users for standard operation (so called built-in users/groups see [Section 3.15.1, "Predefined Users and Groups" \[211\]](#), 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 3.12.3, "LdapUserProvider" \[94\]](#) 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 `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 3.12.3, “LdapUserProvider”](#) [94] 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*.

3.15.6 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` 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@domain`. 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 3.50, "A sample login bouncer configuration"](#) [237]:

```
<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" />
</LoginBouncer>
</LoginBouncers>
```

```
<Pool group="administrator@domain4" min="1" />
</LoginBouncer>
</LoginBouncers>
```

Example 3.50. 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 `domain2`. 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.

3.16 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 - 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 Management-Server* 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 3.2.2, "Oracle Database" \[35\]](#)) and restart the database.

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, Reference \[277\]](#) 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.

CAUTION

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.

CAUTION

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 ("§", "ß", ...). 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`.

Common Attributes

You can add the following attributes to each property:

Property	Description
<code>extensions:translatable</code>	If set to true, marks this property field to be exported to an XLIFF file during translation. See Content Type Model - Properties for Translation [247] for more details.
<code>extensions:automerge</code>	If set to false, marks this property field as not being merged automatically from the master content when the translation task is accepted for a derived content. See Content Type Model - Properties for Translation [247] for more details.

Table 4.1. Common Attributes

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.

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 <i>Utf8Weight</i> 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 <i>Utf8Length</i> and <i>Utf8Weight</i> , <i>Utf8Weight</i> =300 will be used (see below for details) to define the byte length.

Property	Description
<code>extensions:observe</code>	Set to 'true' to make the <code>StringProperty</code> an observed property. This attribute is optional.

Table 4.2. Attributes of a `StringProperty` field

CAUTION

When the `StringProperty` is observed, the length must not exceed 256. Otherwise, the content server cannot start and you will see a log entry like the following in the content server log:

```
2018-12-10 15:01:38 [ERROR] hox.corem.server.Server [] - \
Cannot startup due to malformed document type definition: \
The length 257 of the observed StringProperty 'observed' exceeds the allowed
maximum length 256 (Init Process)
```



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.

Property	Description
<code>Grammar</code>	Defines the DTD or Schema that should be used for this content. <code>core-media-richtext-1.0</code> is the standard grammar for CoreMedia RichText fields. The special predefined grammar <code>coremedia-struct-2008</code>

Property	Description
	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.

Table 4.3. Attributes of an `XmlProperty` field

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 Content Cloud*. 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.

Name	Description
<code>LinkType</code>	If the attribute <i>LinkType</i> 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.
<code>Max</code>	The maximum amount of references for this link property.
<code>extensions:weakLink</code>	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".

Name	Description
	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. Caution! The introduction of weak links to a document type model comes with a caveat: • 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.

Table 4.4. Attributes of `LinkListProperty`

Example for creating a `LinkListProperty`:

```
<LinkListProperty Name="Images" LinkType="Image"/>
```

Only content items of type `Image` or subtypes are allowed in this link list.

NOTE

If you want to create an annotated link list from Blueprint [see Section 7.21, “Annotated LinkLists” in *Studio Developer Manual*], you have to define an XML Struct property, not a `LinkList` property.



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.

Attributes for Translation

The attributes `extensions:translatable` and `extensions:automerge`, which can be attached to all properties, affect the translation behavior. `extensions:automerge` also affects the synchronization behavior.

Both attributes have no effect when you derive a site. A derived site is a complete copy of the master site. Only links are changed, so that they point to content in the derived site and not to content in the master site.

Translatable Properties

In order to support automatic translation processes, properties of the document model can be marked as translatable. When translating a document automatically, 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.

See section "Translatable Predicate" in *Blueprint Developer Manual* for other ways to mark a property as translatable, for example to mark nested properties of a Struct property as translatable.

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 disable automatic merging for a non-translatable property, set the `extensions:automerge` attribute to `false`:

```
<XmlProperty Name="settings" Grammar="coremedia-struct-2008"
  extensions:automerge="false"/>
```

For a synchronization workflow, by default all properties are synchronized between a master content and its derived content. This includes translatable properties. You can disable this behavior by setting `extensions:automerge` to false or by unchecking

the checkbox *Keep synchronized with Master* of the content in Studio (see Section “Removing Content Permanently from Synchronization” in *Studio User Manual* for details).

Note that multi-site properties `master`, `masterVersion`, `locale`, and `ignoreUpdates` are never merged automatically. These properties are configured in the `com.coremedia.cap.multisite.SiteModel`.

If needed, the set of automatically merged properties can be further customized with a custom implementation of interface `AutoMergePredicateFactory`. Such a custom implementation can then be configured in a custom workflow definition for the `AutoMergeTranslationAction`, see Section “AutoMergeTranslationAction” in *Blueprint Developer Manual* for details.

Note, that it is not possible to configure automatic merging of translatable properties for a translation workflow by setting the `extensions:automerge` attribute to `true`. Translatable properties are excluded by the default implementation of the mentioned `AutoMergePredicateFactory`.

Release 2104.3 introduces new configuration options to enable automatic merging of translatable properties, if the values haven't changed in the derived content. This change improves merge results but is not yet the default in this release. It will become the default in future releases. For details, see the release notes and the description of the `workflow.localization.auto-merge.*` configuration properties in Table 4.20, “Workflow Server Properties” in *Deployment Manual*.

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.

NOTE

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.



NOTE

If you are using MySQL, things are much easier. The Content Server will always use the defined length for creating columns in MySQL. Do not specify a `UTF8Weight` or a `UTF8Length` attribute.



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` [see [Section 5.1, “Configuration Property Reference” \[278\]](#) for details].

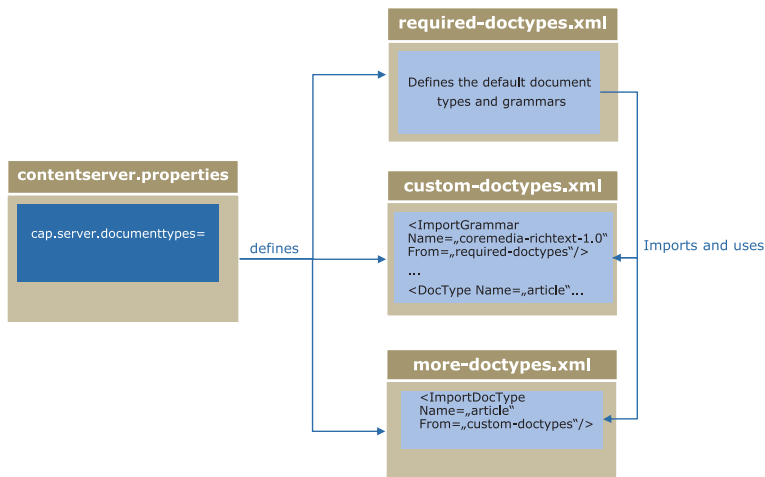


Figure 4.1. Schema of content type definitions

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-doctype.xml` file and are also used in the `specific-doctype.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">

  <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 4.1. The base-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>
```

Example 4.2. The specific-doctypes.xml file

4.2.1 Structure of Content Type Definitions

The basis for your content items in a *CoreMedia CMS* system is the XML file `conf/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 Site Manager Developer Manual and Studio Developer Manual.

CAUTION

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, CmProcessor-Usage, CmRules, CmUserGroup, CmUsers, Dictionary, EditorPreferences, FolderIndex, LinkChangeLog, LinkLists, ObservedValues, ObservedValueChangeLog, MaxIds, MaxMemberIds, MySQLBlobStore, Preferences, Query, QueryIdTable, ReplicatorIdTable, Resources, SgmlData, SgmlGrammar, SgmlText, System, Texts, TrashIdTable, Wf-Definitions, 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 `Document Type` 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"/>
```

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>\$INSTALL_DIR</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.

Property	Description
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.

Table 4.5. Attributes of *XmlGrammar* element

The following `<XmlGrammar>` elements are predefined:

- `coremedia-richtext-1.0`, see `RichtextDtd`
- `coremedia-dictionary`
- `coremedia-query`, see `QueryDtd`
- `coremedia-struct-2008`, see `StructDtd`
- `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 <code>\$INSTALL_DIR</code> which specifies the location of the XML schema in the <i>Content Server</i> 's 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>

Property	Description
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 4.6. 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 4.7. 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” \[250\]](#)). 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” \[251\]](#)).
- Inherit from existing content types (see [Section 4.2, “Creating Content Type Definitions” \[250\]](#)).
- Attach new properties to existing content type definitions.

The first two ways are described in [Section 4.2.1, “Structure of Content Type Definitions” \[251\]](#) and [Section 4.2, “Creating Content Type Definitions” \[250\]](#). 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.

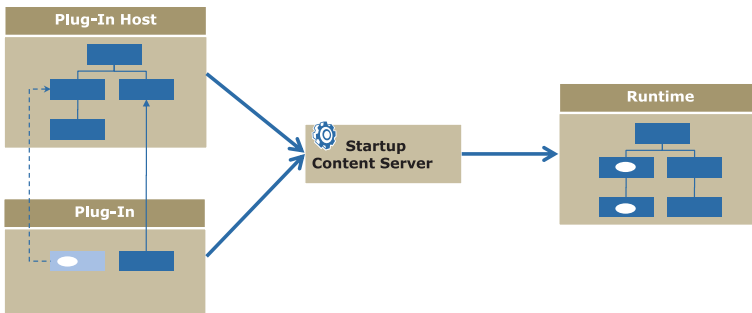


Figure 4.2. Extending content types

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>
```

If you add an `IntProperty`, a `StringProperty` or a `DateProperty` to an existing content type model of a content server after the first start, be sure, that the properties `sql.schema.checkColumns` and `sql.schema.alterTable` file are set to "true".

4.3 Schema Update

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.

CAUTION

Always make a backup of your data before schema update. 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.

- `linklists`: Connects the content item which links with the item to which it links.
- `observedvalues`: Connects the content with an observed property with its value.

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" \[242\]](#) 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.
VERSIONUUID_	Universally Unique Identifier (UUID) of the version.
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").
ISTRANSLATIONAPPROVED_	Marker if this version of the content item is approved for translation ("1") or not ("0").

Attribute	Description
EDITORID_	User ID of the editor.
APPROVERID_	User ID of the approver.
PUBLISHERID_	User ID of the publisher.
TRANSLATIONAPPROVERID_	User ID of the approver for translation.
EDITIONDATE_	Date of the last change.
APPROVALDATE_	Date of the approval.
PUBLICATIONDATE_	Date of the publication
TRANSLATIONAPPROVALDATE_	Date of the approval for translation.

Table 4.8. System attributes of the content type table

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.

The observedvalues table

The primary key of the `observedvalues` table consists of the following properties:

- `SourceDocument`: ID of the content item with an observed property.
- `SourceVersion`: Version of the content item with an observed property.
- `PropertyName`: Name of the observed property.

4.3.2 Struct Properties

Struct properties are special because they might prevent the *Content Server* from restart. This is because struct properties might contain link or link list properties with links which are restricted to specific content types [see [Example 4.6](#), "Struct XML with link list property" [262]].

```
<?xml version="1.0" encoding="UTF-8"?>
<Struct xmlns="http://www.coremedia.com/2008/struct"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <StringProperty Name="string">foo</StringProperty>
  <LinkProperty Name="link"
    LinkType="coremedia:///cap/contenttype/CMArticle"
    xlink:href="coremedia:///cap/content/2"/>
  <StructProperty Name="substruct">
    <Struct>
      <LinkListProperty Name="links"
        LinkType="coremedia:///cap/contenttype/Content_">
        <Link xlink:href="coremedia:///cap/content/2"/>
        <Link xlink:href="coremedia:///cap/content/102"/>
      </LinkListProperty>
    </Struct>
  </StructProperty>
</Struct>
```

Example 4.6. Struct XML with link list property

If you change the content type name, these links are invalid and you get an error.

```
java.lang.NullPointerException
at
com.coremedia.cotopaxi.struct.xml.StructUnHandler.unparseLinkList (StructUnHandler.java:194)
```

Example 4.7. Error message because of invalid link type

4.3.3 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`.
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.

4.3.4 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. So you have to do some changes in the content types definition files and in the database.

Note that you may also have to adapt various parts of your application, such as Queries stored in the repository, link list properties in Structs, template file names and EL expressions in templates, generated ContentBeans etc.

In the following, you find a description to consistently rename content types in the `doctypes.xml` file and in the database and to update Struct properties.

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 `LinkType` attribute of all `LinkListProperties`.
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. If you have Struct properties with link lists that reference the renamed content type, you have to adapt the `LinkType`. In principle, you have several options to do the update of the `LinkType`. However, the one described here is the one which introduces the least downtime. The update consists of the following steps:
 - a. Add a new abstract content type with no properties to the content type definition file. Give the content type the old name of the renamed type and also the same parent as the renamed type.
 - b. Add the new abstract content type as the parent type to the renamed content type.
 - c. Restart the Content Server. Everything should work as before. That is, the Content Server won't fail to start and your editors may continue to work. As the intermediate content type is abstract, they won't be able to accidentally create contents of this type.
 - d. Change the `LinkType` of the Struct properties to the new content type. If you have content queries, also change the content type if necessary. You can write custom UAPI clients for these tasks.
 - e. When you have finished the update, remove the new content type from the content type definition file.
8. Restart the *Content Servers*.

Adapting the ChangeLog

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 3.11, "Truncate the ChangeLog" \[83\]](#).
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.

Existing content of the renamed type needs to be reindexed in *Content Feeder* and *CAE Feeder* indices. For the *Content Feeder*, use the JMX operation `reindexByType` of *ContentFeederMean.com.coremedia:type=AdminBackgroundFeed, application=content-feeder* with the new name of the content type as first parameter and value `type` as second parameter ("aspect") to reindex the contents. For the *CAE Feeder*, you need to reindex from scratch. See Section 3.5, "Reindexing" in *Search Manual* for details.

4.3.5 Deleting Content Types

CAUTION

Before you delete content types, make sure that the data is no longer needed.



Preliminary Considerations

Recommended Approach

Deletion of content types is a challenging task, mostly because of structs (see [Section 4.3.2, "Struct Properties" \[262\]](#)) and content versions:

- A struct might have a link property to the to-be-deleted content type. If this link property is not used (= does not link to an actual content), it is hard to find and change. If you delete the content type and start the server, you will get problems.
- Older versions of a content might link to the to-be-deleted content type. You can use the *cleanversions* tool (see [Section "Clean Versions" \[140\]](#)) to delete older versions, but the tool makes some exceptions which complicate the task, especially for newly referenced versions.

Due to the aforementioned challenges, CoreMedia recommends not deleting the content type immediately, but to follow these steps:

1. Discontinue the use of the content type in your team.
2. Disable the creation of the content type in *Studio* (see Section 7.3.6, "Excluding Document Types from the Library" in *Studio Developer Manual*) and disable the selection of the content type in the struct editor.
3. Delete content items of the to-be-deleted content type from the repository. Follow the description in [section "Deleting content from the repository" \[266\]](#).
4. Wait a few weeks or more for sufficient editorial changes, so that more and more versions referring to outdated content type are replaced by new ones. The longer you wait, the easier the subsequent steps are.

5. Use *cleanversions* to delete older versions which might still refer to the to be deleted content type. Follow the steps described in [section "Deleting older version with cleanversion" \[266\]](#).
6. Delete the content type and all references from the content type definition file and the database. Follow the description in [section "Deleting content type from database and content type definition" \[267\]](#).

Deleting content from the repository

*Delete content type
from the content repos-
itory*

Delete all content items of the to-be-deleted content type from the content repository.

1. Use a JShell script or a UAPI client to select all content items of the content type to delete.
2. Find all referrers to this content (from link lists, richtext fields, Structs...) and decide what to do (delete the link, link to other content ...). For link lists you can use the *validate-link-type* tool (see [Section "Validate Link Type" \[112\]](#)).
3. Find all link or link list properties in struct properties whose link type is the to-be-deleted content type. Change the link type or remove the properties.
4. Publish all changed content, so that no old version is the latest published version.
5. Withdraw all to-be-deleted content.
6. Delete the content.

Deleting older version with cleanversion

Delete older versions

At this point, you have deleted all content items of the to-be-deleted content type from the repository. No other current content item links to this type. However, there might be older versions of content items which still link to the type.

You can use *cleanversions* to delete these older versions. However, all content versions which are still in use will not be deleted (read the section about default exclusions in [Section "Clean Versions" \[140\]](#)).

1. Run `cm cleanversions` (see [Section "Clean Versions" \[140\]](#)) to clear any unused versions. If you do not want to apply it globally, search for all document versions containing links in Struct properties whose link type is the to-be-deleted content type. Ensure to use `--keep-number=1` to delete as many versions as possible.
2. Validate content state using UAPI tooling to see contents/versions which are still blocking deletion.
3. When there are still blocking content/version, consider the following options:
 - Especially in multi-site scenarios you might have to abort existing translation workflows and trigger a new translation workflow with the current content version.

- Wait some more time to get rid of the blocking versions and start again with step 1.
- Decide on alternative actions to get rid of remaining contents/versions.

Deleting content type from database and content type definition

Delete content type from database and definition file

Now, you have removed all content items of the to-be-deleted type from the repository and no other items link to the type. Delete the last occurrences of the type in rules and from the recycle bin and ultimately from the content type definition file.

1. Delete all rules that are attached to the document type by using the *User Manager* or by setting the following SQL statement (stop the Content Servers before applying the SQL statement). You have to apply the SQL statement at all Content Server databases:

```
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.

2. Use the `cleanrecyclebin` utility to remove all the content items from the recycle bin.
3. Stop the *Content Management Server* and the *Live Servers*.
4. Remove the content types to delete from the content types definition files of all *Content Servers*. Change the `LinkType` property of link list properties to a new content type.
5. Verify that the table which represents the content type is empty (see [Section 4.3.1, "The Database Schema" \[259\]](#) for details about the database schema).
6. Drop the table which represents the content type from the databases.
7. Start the *Content Servers* again.

4.3.6 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` file are set to "true"
4. Restart the *Content Server*.

4.3.7 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 `doctypes.xml` file and in the database.

Generally, all *Content Servers* should be updated 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 update 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:

NOTE

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. When the property is observed, execute the following SQL statement with the declaring content type and all its subtypes:

```
UPDATE ObservedValues SET propertyName = 'myNewName'
WHERE propertyName='myOldName' AND EXISTS
(SELECT * FROM Resources WHERE id_=sourceDocument
AND documentType IN
('myType', 'mySubtype1', 'mySubtype2',...))
```

5. 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*.

4.3.8 Changing Properties

This section describes some use cases for property changes.

Changing Property Type

Changing the type of a property (for example from `StringProperty` to `XMLProperty`) is only sensible when the property contains no content. However, you can not simply change the type of an existing property. Instead, you have to delete the property [see [Section 4.3.9, “Deleting Properties” \[274\]](#)] and afterwards introduce the new property [see [Section 4.3.6, “Adding Properties” \[267\]](#) or use aspects to add properties, see [Section 4.2.3, “Attaching Properties to Existing Content Types” \[257\]](#)] into the content type model.

Changing the Size of a StringProperty

Sometimes it might be necessary to change the size of a string property. For example, when the size exceeds a database limit.

Prerequisites

Be sure that you have a backup of your data. When you want to reduce the length, make sure that all existing content fits into the new size.

CAUTION

When the `StringProperty` is observed, the length must not exceed 256. Otherwise, the content server cannot start and you will see a log entry like the following in the content server log:

```
2018-12-10 15:01:38 [ERROR] hox.corem.server.Server [] - \
Cannot startup due to malformed document type definition: \
The length 257 of the observed StringProperty 'observed' exceeds the allowed
maximum length 256 (Init Process)
```



Proceed as follows:

1. Be sure to have a backup of your data.
2. Stop the Content Server.
3. Change the `Length` attribute to the new length.
4. Make sure that property `sql.schema.alterTable` is set to true.
5. Start the Content Server again.

You should see log entries like the following in the content server log.

```
2018-08-22 13:33:11 [INFO] hox.corem.server.sql.SQLStore [] - \
SchemaValidator: validating document types (Init Process)
2018-08-22 13:33:11 [INFO] hox.corem.server.sql.SQLStore [] -
DocumentTypeRegistry: altering table: \
ALTER TABLE CMAAction MODIFY "teaserTitle" VARCHAR(300) (Init Process)
2018-08-22 13:33:11 [INFO] hox.corem.server.sql.SQLStore [] -
DocumentTypeRegistry: altering table: \
ALTER TABLE ESDynamicList MODIFY "teaserTitle" VARCHAR(300) (Init Process)
```

Example 4.8. Log entries when string property has been changed

Changing the Observe Attribute of a StringProperty

NOTE

The length of an observed `StringProperty` must not exceed 256.



When an existing `StringProperty` changes its `observe` attribute then the values of the observed property in the table `ObservedValue` will be automatically added or removed. In the following SQL statements `CMEExternalCategory` is the example name of the content type and `externalId` the example name of the property.

Proceed as follows:

1. Be sure to have a backup of your data.
2. Stop the Content Server.
3. When the not observed `StringProperty` property should be observed, change the `observe` attribute in the content type definition file to 'true'.

When the already observed `StringProperty` property should not be observed anymore, remove the `observe` attribute.

4. Start the Content Server again.

Check the logs

When the property is newly observed, the existing values of the property will be automatically inserted into the `ObservedValues` table. You should see log entries like the following in the Content Server log.

```
2018-08-22 13:33:11 [INFO] hox.corem.server.sql.SQLStore [] - \
DocumentTypeRegistry: Update observed property:
The property 'externalId' of the document type 'CMEExternalCategory' is newly
```

```
observed. The following command is being executed: \
INSERT INTO ObservedValues (sourceDocument, sourceVersion, propertyName,
propertyName)
SELECT c.id, c.version, 'externalId', c."externalId" FROM Resources r,
CMExternalCategory c
WHERE r.id = c.id AND r.latestVersion = c.version AND c."externalId" IS
NOT NULL AND c."externalId" != '' ORDER BY r.id
2018-08-22 13:33:12 [INFO] hox.corem.server.sql.SQLStore [] - \
DocumentTypeRegistry: The statement took 1000 ms and 1000 rows were affected:
INSERT INTO ObservedValues (sourceDocument, sourceVersion, propertyName,
propertyName)
SELECT c.id, c.version, 'externalId', c."externalId" FROM Resources r,
CMExternalCategory c
WHERE r.id = c.id AND r.latestVersion = c.version AND c."externalId" IS
NOT NULL AND c."externalId" != '' ORDER BY r.id
```

Example 4.9. Log entries when an existing string property is newly observed

When the property isn't any longer observed, the values of the property will be removed from the `ObservedValues` table. You should see log entries like the following in the content server log.

```
2018-08-22 13:33:11 [INFO] hox.corem.server.sql.SQLStore [] - \
DocumentTypeRegistry: Update observed property:
The property 'externalId' of the document type 'CMExternalCategory' is not
observed anymore. The following command is being executed: \
DELETE FROM CMExternalCategory WHERE propertyname = 'externalId' AND
sourceDocument = (SELECT DISTINCT id_ FROM CMExternalCategory WHERE id_ =
sourceDocument)
2018-08-22 13:33:12 [INFO] hox.corem.server.sql.SQLStore [] - \
DocumentTypeRegistry: The statement took 1000 ms and 1000 rows were affected:
DELETE FROM CMExternalCategory WHERE propertyname = 'externalId' AND
sourceDocument = (SELECT DISTINCT id_ FROM CMExternalCategory WHERE id_ =
sourceDocument)
```

Example 4.10. Log entries when an existing string property is not observed anymore

Changing LinkType of LinkListProperty

There are two different cases when changing the *LinkType* attribute of a `LinkListProperty` definition.

Choosing a supertype of the current LinkType

If the new *LinkType* is a supertype of the current *LinkType*, it's sufficient to restart the *Content Servers* after changing the document type definition.

Choosing a subtype of the current LinkType or another unrelated type

If the new *LinkType* is not a supertype of the current *LinkType*, then links in existing content may point to documents of wrong types. Before changing the content

type definition, check whether such content exists by using the command-line tool `cm validate-link-type`, which is described in [Section “Validate Link Type” \[112\]](#)

For example, if you want to change the *LinkType* attribute for property `pictures` of type `Teasable` from `Media` to subtype `Picture`, check if there's any link that would violate the new *LinkType*:

```
cm validate-link-type -u admin -t Teasable -l pictures -n Picture -a
```

The example uses the `-a` option to also check old document versions. Because old versions can easily be restored by editors, they should be valid as well.

If the tool does not find any invalid links for the new link type, change the *LinkType* attribute in the document type definition and restart the *Content Servers*.

If the tool finds existing content with links that would be invalid for the new link type, you have different options:

1. Decide to not change the *LinkType*.
2. Remove documents or document versions with invalid links from all servers. Consider using tools such as [Section “Clean Versions” \[140\]](#) or [Section “Destroy” \[194\]](#).
3. If really necessary, it's possible to remove links from the `linklists` database table for all *Content Servers* after they have been stopped. See [Section 4.3.1, “The Database Schema” \[259\]](#) for a description of the database table. Database changes must be done with extreme caution. If you delete rows from the `linklists` table, you must not forget to update the value of the `LinkIndex` column for other affected rows. The `LinkIndex` column represents a link's position in a link list. Therefore, its value must be reduced for all links after you removed a link in the same property of the same document version.

4.3.9 Deleting Properties

CAUTION

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> If the property is observed, remove the observed property information from the <code>observedvalues</code> 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. For example: Delete the <code>StringProperty</code> named <code>ExternalId</code> from the content type <code>ExternalProduct</code>. <pre>DELETE FROM ObservedValues WHERE propertyname = 'ExternalId' AND sourceDocument = (SELECT DIS TINCT id_ FROM ExternalProduct WHERE id_ = sourceDocument);</pre> <pre>COMMIT;</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 Art icle WHERE id_ = documentid);</pre> <pre>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.

Property type	How to change
	For example: Delete the XmlProperty named Text from the content type Article. <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 DISTINCT id_ FROM Article WHERE id_ = sourceDocument); COMMIT;</pre>
LinkListProperty	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. For example: Delete the LinkListProperty named RelatedDocuments from the content type Article. <pre>DELETE FROM LinkLists WHERE propertyname = 'RelatedDocuments' AND sourceDocument = (SELECT DISTINCT id_ FROM Article WHERE id_ = sourceDocument); COMMIT;</pre>

Table 4.9. How to delete properties of different type

5. Reference

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.

Properties can now be configured via Spring Boot. You can use `application.properties`, system properties, environment variables in uppercase and many more. See <https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html> for details.

Spring Boot configuration

Spring's relaxed binding also allows for different notations of property names like snake or camel case, but the default is 'kebab case' (separating words with dashes). Generally a dot in a property name reflects some kind of logical hierarchy. List-valued properties are zero-based and use bracket notation (`x.y.1.*` -> `x.y[0].*`)

Spring relaxed binding

5.1 Configuration Property Reference

Different aspects of the *Content Server* can be configured with properties. All configuration properties are bundled in the Deployment Manual [Chapter 4, *CoreMedia Properties Overview* in *Deployment Manual*]. The following links reference the properties that are relevant for the *Content Server*:

- Section 4.2, “Content Server Properties” in *Deployment Manual* contains properties for the general configuration of the *Content Server*.
- Section 4.2.2, “CORBA Properties” in *Deployment Manual* contains properties for the CORBA connection of the *Content Server*.
- Section 4.2.3, “Properties for the Publisher” in *Deployment Manual* contains properties for the configuration of the publisher component of *Content Server*.
- Section 4.2.4, “Properties for the Connection to the Database” in *Deployment Manual* contains properties for the configuration of the SQL database connection.
 - Connection with 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.
 - How the *Content Servers* deal with content type changes. All checks and changes are performed at starting time of the *Content Servers*.
 - Since *SCI 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).
 - Configuration of the blob and SgmlText Collector. The Collector is a component which erases SgmlText and blobs from content versions which have been deleted.
 - Configuration of XML caching
 - Configuration of the SQL connection pool
- Section 4.2.5, “Properties for Replicator Configuration” in *Deployment Manual* contains properties for the configuration of the replicator component in *Replication Live Servers*.
- Section 4.2.6, “Properties for Timezone and IOR” in *Deployment Manual* contains properties for the configuration of the timezone and the location of the CORBA IOR.
- Section 4.2.7, “Renamed Properties” in *Deployment Manual* contains an overview of old and new names of renamed *Headless Server* properties.

- Section 4.10, “UAPI Client Properties” in *Deployment Manual* contains properties for UAPI clients which can also be used by the *Headless Server*.

5.2 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.

NOTE

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>JavaClasspath</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.

Attribute	Type	Description
<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.cache.resource-cache-size</code> .
<i>ResourceCacheEntries</i>	Read-only	Number of resources entered into the resource cache in the last <code>ResourceCacheInterval</code> .
<i>ResourceCacheEvicts</i>	Read-only	Number of cache evicts in <code>ResourceCacheInterval</code> .
<i>ResourceCacheHits</i>	Read-only	Number of cache hits in <code>ResourceCacheInterval</code> .
<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: offline, maintenance, administration, or online.
<i>RunLevelNumeric</i>	Read-only	The current run level of the Content Server: 0=offline, 1=maintenance, 2=administration, or 3=online.
<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 <code>SwitchRunLevel</code> grace period. "-" 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 highest <code>RepositorySequenceNumber</code> with a Replication Live Server's <code>LatestIncomingSequenceNumber</code>

Table 5.1. JMX manageable attributes of the Content Server

Content Server Operations

Operation	Parameter	Description
<i>switchToRun- Level</i>	p1 - Runlevel to switch to	Change the run level of the Content Server. Possible run levels are:
	p2 - Delay before switching in seconds	• offline • maintenance • administration • online
<i>abortRun- LevelSwitch</i>		Stop switching to another runlevel.

Table 5.2. JMX manageable operations of the Content Server

Publisher Attributes

Attribute	Type	Description
AvPublPrevTime	Read-only	Average publication preview time [msec] per publication preview.
AvPublSize	Read-only	Average publication size
AvPublTime	Read-only	Average publication time [msec] per publication.
AvWaitTime	Read-only	Average waiting time [msec] in the queue.
DestroyIntVersions	Read/Write	Deprecated flag for destroying intermediate content versions. Use DestroyIntermediateVersions instead. "true" maps to mode "STRICT" and "false" maps to the mode "OFF".
DestroyIntermediateVersions	Read/Write	Destroy intermediate content versions. Values are "OFF", "STRICT" and "DUMB"
DestroyOlderVersionsOn-LiveServers	Read/Write	Keep only the newest version on the Live Server. Values are "true" and "false".

Attribute	Type	Description
EnableBypassPreviews	Read/Write	Whether publication previews bypass ("true") or not ("false") the publication queue for faster response times.
FailedPublCount	Read-only	Number of failed publications.
FailedPublPrevCount	Read-only	Number of failed publication previews.
LastPublDate	Read-only	Date of the last publication in Unix format (milliseconds since 01.01.1970).
LastPublDate	Read-only	Date of the last publication in human readable format.
LastPublResult	Read-only	Result of the last publication (success or failed).
LastPublSize	Read-only	Size of the last publication (in resources).
LastPublTime	Read-only	Required time for last publication in [msec].
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

Attribute	Type	Description
PublicationTargetNames	Read-only	Names of the publication targets if Multi-Master publishing is used.
QueueSize	Read-only	Queue size of the Publisher (in content items).

Table 5.3. JMX manageable attributes of the Publisher

Multi-Master 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 5.4. JMX manageable attributes for Publication Targets

Publisher Operations

Operation	Parameter	Description
<code>setPriority</code>	p1 - Client p2 - priority	Set the priority for the defined client. This operation sets the properties <code>publisher.priority.<client></code> . The priority is an integer value from 0 to 100 and the client is an integer value from the following list: • 0 - The editor GUI • 1 - The Unified API

Operation	Parameter	Description
		• 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 5.5. 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.
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.

Attribute	Type	Description
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).

Table 5.6. JMX manageable attributes of the Replicator

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)	The <i>Content Application Engine</i> [CAE] is a framework for developing content applications with <i>CoreMedia CMS</i>. While it focuses on web applications, the core frameworks remain usable in other environments such as standalone clients, portal containers or web service implementations. The CAE uses the Spring Framework for application setup and web request processing.
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 CoreMedia CMS content items or to any other kind of third-party systems. Various CoreMedia components like the CAE Feeder 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	The <i>Content Delivery Environment</i> is the environment in which the content is delivered to the end-user. It may contain any of the following modules: • <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> • <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: • <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 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: <i>Content Servers</i> are web applications running in a servlet container. • <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]	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. CORBA programs communicate using the standard IIOP protocol.

Glossary |

CoreMedia Studio	<i>CoreMedia Studio</i> is the working environment for business specialists. Its functionality covers all the stages in a web-based editing process, from content creation and management to preview, test and publication. 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.
Dead Link	A link, whose target does not exist.
DTD	A Document Type Definition is a formal context-free grammar for describing the structure of XML entities. The particular DTD of a given Entity can be deduced by looking at the document prolog: <pre><!DOCTYPE coremedia SYSTEM "http://www.coremedia.com/dtd/coremedia.dtd"</pre> 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.
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 used in former CoreMedia Studio version for the declarative development of complex Ext JS components. EXML is Jangaroo 2's equivalent to Apache Flex (formerly Adobe Flex) MXML and compiles down to ActionScript. Starting with release 1701 / Jangaroo 4, standard MXML syntax is used instead of EXML.
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.
Headless Server	CoreMedia Headless Server is a CoreMedia component introduced with CoreMedia Content Cloud which allows access to CoreMedia content as JSON through a GraphQL endpoint. The generic API allows customers to use CoreMedia CMS for headless use cases, for example delivery of pure content to Native Mobile Applications, Smart-watches/Wearable Devices, Out-of-Home or In-Store Displays or Internet-of-Things use cases.
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 <i>CAE</i> . If you are using the <i>CoreMedia Multi-Master 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.
MXML	MXML is an XML dialect used by Apache Flex (formerly Adobe Flex) for the declarative specification of UI components and other objects. CoreMedia Studio uses the Open Source compiler Jangaroo 4 to translate MXML and ActionScript sources to JavaScript that is compatible with Ext JS 6.
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	In relation to CoreMedia, properties have two different meanings: 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 item depends on the content type. In connection with the configuration of CoreMedia components, the system behavior of a component is determined by properties.
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	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. A typical site also has a master site it is derived from.
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. The Site Manager is deprecated for editorial use.
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	In CoreMedia, JSPs used for displaying content are known as Templates. OR

	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, predefined layout and even predefined content.
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	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. Caution! Weak links may cause dead links in the live environment.
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, 256
 administrator group, 232
 administrator groups, 232

B

BlobProperty, 242
 blobProperty, 245

C

change password
 system user, 216
 cm
 content-uuid-export, 147, **151**
 content-uuid-import, 147, **155**
 dumpusers, **158**
 generate-content-uuid-map, 147, **153**
 restoreusers, **175**
 serverexport, 180, **184**
 serverimport, 180, **182**
 validate-link-type, **112**
 validate-multisite, **114**
 CollectablePredicate, 140
 configuration, 60, 63
 Content Management Server Group, 233
 Content Server
 properties, 278
 content server
 JMX, 280
 content server groups, 233
 content type
 changing string observe, 272
 changing string size, 271
 content types
 abstract type, 256
 adding, 262
 adding properties, 267

changing LinkType, 273
 deleting, 265
 grammar, 253
 inheriting, 256
 name limitations, 252
 names, 252
 properties, 242
 renaming, 263
 renaming properties, 268
 schema update, 259

CoreMedia CMS, 25
 creationDate, 28

D

database, 35, 37, 39
 dateProperty, 244
 deleting content types, 265
 document types
 creation, 250
 deleting properties, 274
 extending, 257
 DocumentTypes, 241

E

editingDate, 28

G

garbage collection, 137
 Grammar, 251
 grammar, 253
 groups
 administrator groups, 232
 predefined, 211
 workflow groups, 215

I

indexing, 246
 intProperty, 243

J

JMX, 280
 JMX management, 210
 JShell, 161
 .cm.jshell.properties, 164
 /exit, 167
 /save, 167

- _cm.jshell.properties, 164
- cm jshell, 161
- cm namespace, 162
- coremedia namespace, 162
- COREMEDIA script, 162
- coremedia.close[], 167
- coremedia.connect[], 167
- coremedia.connection, 166
- coremedia.help[], 162, 165
- Environment Variables, 164
- Examples, 165
- Exit Code, 167
- Initial Connection Configuration, 162
- Password Prompt, 165
- System Properties, 163
- User Properties, 164

L

- LDAP
 - user changes, 99
- LDAP integration, 85
- ldaps, 98
- link
 - weak, 245
- linkListProperty, 245
- Live Server Group, 234
- live server groups, 234

M

- managed properties, 280
- modificationDate, 28
- multi-master management, 23
 - migrate to, 74

O

- observe, 244

P

- properties, 14
 - blobProperty, 245
 - configuration, 277
 - dateProperty, 244
 - indexing, 246
 - intProperty, 243
 - linkListProperty, 245
 - maximum name length, 242

- semantics, 277
 - stringProperty, 243
 - types, 242
 - UTF-8 encoding, 248
 - xmlProperty, 244
- publicationDate, 28

R

- Replication Live Server, 15, 17-18, 60, 62, 67
- rights
 - computation of, 227
- runlevel, 53

S

- schema update, 259
- Search Engine, 1
- server utilities, 101, 186
 - cm content-uuid-export, 147, **151**
 - cm content-uuid-import, 147, **155**
 - cm dumpusers, **158**
 - cm generate-content-uuid-map, 147, **153**
 - cm groovysh, 168
 - CM IOR, 105
 - cm jshell, 161
 - cm restoreusers, **175**
 - Groovy Shell, 168
 - JShell, 161
 - serverexport, 180, **184**
 - serverimport, 180, **182**
- serverexport, 180, **184**
- serverimport, 180, **182**
- stringProperty, 243
- system user
 - change password, 216

T

- troubleshooting, 19-20, 22, 239
 - wrong files in My Edited Content, 100

U

- users, 211
 - assigning licenses, 235
 - predefined, 211
 - rights management, 217
- UUID
 - Content, 147, 181, 183, 185

V

validate-link-type, **112**

validate-multisite, **114**

W

weak link, 245

workflow rule groups, 215

X

xmlProperty, 244