COREMEDIA CONTENT CLOUD

CoreMedia Content Cloud v12 Upgrade Guide

COREMEDIA

List of Tables

1. Preface

This manual describes the upgrade from a *CoreMedia CMCC v11 Blueprint* system to a *CoreMedia CMCC v12 Blueprint* system.

- Chapter 2, *CoreMedia Content Cloud v12* [13] gives a short overview of the main changes in *CoreMedia Content Cloud* v12. It describes some of the benefits you get from your upgrade tasks.
- Chapter 3, *Upgrade Information Overview* [16] describes the actual upgrade tasks in general. The following chapters then go into more detail per application and per feature.
- Chapter 4, *Prerequisites for the Upgrade* [18] describes the development environment and the software you need to have for building the workspace.
- Chapter 5, *Changes in Supported Environments* [20] describes changes in the supported environments, such as databases.
- Chapter 6, *Removed Components and Features* [24] lists the removed components and features and links to more information.
- Chapter 7, *Detailed Overview of Changes* [30] gives a high-level overview of all changes. This includes new, deprecated and removed components and third-party software; new operation and configuration schemes, and much more.

1.1 Audience

This manual is intended for developers, architects, and project managers who plan to upgrade from *CoreMedia Content Cloud* v11 to *CoreMedia Content Cloud* v12.

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

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

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.
ϟ	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” [7] 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-12

Refer to our Blueprint Github mirror repository for recommendations to upgrade the workspace either via Git or patch files. You will also find how-tos for upgrading the system on our documentation website.

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 parts of 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* .

npm packages

CoreMedia provides parts of its release artifacts as npm packages under the following URL:

https://npm.coremedia.io

Your pnpm client first needs to be logged in to be able to utilize the registry (see 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, how-tos and Javadoc as PDF files and as online documentation at the following URL:

https://documentation.coremedia.com

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 *Adaptive Personalization*, the CoreMedia module for personalized websites, and Client-Side Personalization. You will learn how to configure the GUI used in *CoreMedia Studio*, how to use predefined contexts and how to develop your own extensions.
Analytics Connectors Manual	Developers, architects, administrators	This manual describes how you can connect your CoreMedia website with external analytic services, such as Google Analytics.
Blueprint Developer Manual	Developers, architects, administrators	This manual gives an overview over the structure and features of *CoreMedia Content Cloud*. It describes the content type model, the *Studio* extensions, folder and user rights concept and many more details. It also describes administrative tasks for the features. It also describes the concepts and usage of the project workspace in which you develop your CoreMedia extensions. You will find a description of the Maven structure, the virtualization concept, learn how to perform a release and many more.
Connector Manuals	Developers, administrators	This manuals 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.

Manual	Audience	Content
Content Application Developer Manual	Developers, architects	This manual describes concepts and development of the *Content Application Engine (CAE)*. You will learn how to write Freemarker templates that access the other CoreMedia modules and use the sophisticated caching mechanisms of the CAE.
Content Server Manual	Developers, architects, administrators	This manual describes the concepts and administration of the main CoreMedia component, the *Content Server*. You will learn about the content type model which lies at the heart of a CoreMedia system, about user and rights management, database configuration, and more.
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 *Elastic Social* 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 *Headless Server*. 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 *Importer* application and how you define the transformations that convert your content into CoreMedia content.

Manual	Audience	Content
Multi-Site Manual	Developers, Multi-Site Administrators, Editors	This manual describes different otions to desgin 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.
Search Manual	Developers, architects, administrators	This manual describes the configuration and customization of the *CoreMedia Search Engine* and the two feeder applications: the *Content Feeder* and the *CAE Feeder*.
Studio Developer Manual	Developers, architects	This manual describes the concepts and extension of *CoreMedia Studio*. You will learn about the underlying concepts, how to use the development environment and how to customize *Studio* to your needs.
Studio User Manual	Editors	This manual describes the usage of *CoreMedia Studio* for editorial and administrative work. It also describes the usage of the *Adaptive Personalization* and *Elastic Social* GUI that are integrated into *Studio*.
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 *CoreMedia Unified API*, which is the recommended API for most applications. This includes access to the content repository, the workflow repository and the user repository.
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.

Manual	Audience	Content
Workflow Manual	Developers, architects, administrators	This manual describes the *Workflow Server*. This includes the administration of the server, the development of workflows using the XML language and the development of extensions.

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 on our online learning platform (CoreMedia Enablement, live online or at your own location.

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

http://www.coremedia.com/training

Contact the training department at the following email address:

Email: training@coremedia.com

1.3.5 CoreMedia Support

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

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

http://support.coremedia.com/

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

Email: support@coremedia.com

Create a support request

Support request

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

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

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

Support checklist

To put it in a nutshell, CoreMedia needs:

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

Log files

An essential feature for the CoreMedia system administration is the output log of Java processes and CoreMedia components. They're often the only source of information for error tracking and solving. All protocolling services should run at the highest log level that is possible in the system context. For a fast breakdown, you should be logging at debug level. See Section 4.7, "Logging" in *Operations Basics* for details.

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, application containers only write logs to the console output but can be accessed from the container runtime using the corresponding command-line client.

For the *docker* command-line client, logs can be accessed using the **docker logs** command. For a detailed instruction of how to use the command, see docker logs. Make sure to enable the timestamps using the `--timestamps` flag.

```
docker logs --timestamps <container>
```

For the *kubectl* command-line client in a Kubernetes environment you can use the **kubectl logs** command to access the logs. For a detailed instruction of how to use the command, see kubectl logs. Make sure to enable the timestamps using the `--timestamps` flag.

```
kubectl logs --timestamps <pod>
```

2. CoreMedia Content Cloud v12

This guide is intended to help you during your upgrade to CoreMedia Content Cloud 12.2404.1. Familiarize yourself with the changes and actual tasks when upgrading from *CoreMedia Content Cloud* v11 to v12. Use this introduction to learn about the major enhancements and to understand the big picture.

Use the following chapters to review all the information about the changes that concern you – your individual upgrade – in detail.

There's no need to read this guide from A to Z as the chapters can be read individually. Use the search function to navigate quickly to the topics in this guide that are especially interesting for you.

New Features - The Highlights

We have been working on our Studio's interface, releasing various improvements that have the goal to significantly support the users in their daily work and to make it more enjoyable.

Among them:

- Visual refresh of CoreMedia Studio>

  Since design itself is a never ending task, as design evolves permanently, every state-of-the-art-design becomes outdated after time. The new visual update respects these developments and provides the user with higher contrasts that makes it easier to distinguish individual components from one another. The new color pattern increases usability and by adding illustrations and animations to the software, we follow the current trends and give the user a more emotional based feedback to their actions. Basic components have been given a totally new styling with more whitespace, larger fonts and more distinguishable patterns.

- CoreMedia Studio for CMCC 12 on Cloud contains in-app guides>

  CoreMedia Studio is a very powerful tool for content management. To support you in your daily work, we have added in-app guides to help you to get started with CoreMedia Studio and to use it more efficiently. In-app guides describe components of the user interface and provide step-by-step instructions for common tasks.

- Integration with Engagement Cloud

With CoreMedia Engagement Cloud, you can now use segmentation to target content based on profiles and segments. When you create segmented content in Studio, you will see all the segments and profiles from your Engagement Cloud account.

- Support for blobs larger than 2GB

  CoreMedia Studio now supports blobs larger than 2GB in upload and handling. This is especially useful for video content.

Technical Changes - Overview

Besides these new features, CoreMedia has done a lot of work behind the scenes. Updated libraries, an improved developer experience, removal of outdated functionalities, and more. While these improvements will ease your life in the long run, they require immediate upgrade tasks.

The following list shows the most important changes in terms of the upgrade effort, while Chapter 7, *Detailed Overview of Changes* [30] lists all changes in detail. See the http://bit.ly/cmcc-12-supported-environments document for the list of the supported databases, operating systems, and browsers of this release.

- Public API for Studio Client Apps

  The Studio Client API is now public and documented. This allows you to build your own Studio Client Apps and integrate them into CoreMedia Studio. You can find the documentation in the Studio Developer Manual.

- Upgrade to Spring Boot 3.2

  Spring Boot 3.2 is the current version of the Spring Framework extension to build production ready applications using Spring Framework 6. The upgrade to Spring Boot 3.2 was necessary because of the shortened Support Lifecycle of Spring Boot.

- Removal of JSP support

  JSP is an older technology no longer recommended by Spring Boot. The shift away from JSP, in favor of more modern template engines, is driven by the desire for cleaner code, improved separation of concerns, better tooling support, and an enhanced developer experience in the ever-evolving landscape of web development. Therefore, JSP support was removed for CMCC 12. Use Freemarker Templates instead.

  Contact us if your project still requires JSP support.

- Upgrade to Java 17

  Upgrading to Spring Boot 3.2 and Springframework 6.1 required upgrading to Java 17 as the minimum Java version as described in the official Springframework documentation.

- CKEditor 4 no longer supported

  In CMCC 11, Studio switched to CKEditor 5 for rich-text editing; however, CKEditor 4 was still supported. In CMCC 12, CKEditor 4 is no longer supported.

- Removal of Site Manager

Site Manager has been around for many years but has been gradually replaced by Studio. Moreover, as most of the new features over the last years have been implemented for Studio and not for Site Manager, it was time to remove Site Manager. Contact us if you feel that your project still requires Site Manager for some specific Content or Workflow Operations.

3. Upgrade Information Overview

CoreMedia upgrade information is divided into general information and specific upgrade tasks for each component. You can find the general information on our documentation site while the detailed upgrade tasks are part of this manual.

CoreMedia strongly recommends that you read the general information before you start the upgrade.

General Upgrade Information

Have a look at our Upgrade how-to section on the documentation site for general information on how to upgrade your workspace.

- CoreMedia Release Cycle and LTS Strategy describes the CoreMedia release cycle and the long-term support strategy.
- Why You Should Perform an Upgrade gives you technical and business reasons for upgrading your workspace.
- How to Manage Upgrade Tasks gives you hints on how to manage the upgrade tasks.
- General Upgrading Tasks describes the general steps you have to perform when upgrading your workspace. For example, upgrading to the last version of your current version, removing outdated components, and merging the new version of the CoreMedia Blueprint.
- Upgrading with Git describes in general terms, how you can merge the new version of the CoreMedia Blueprint into your existing workspace.
- Upgrading Guides leads you to all existing upgrade guides, for example, to this one.

Specific Upgrade Information

In this manual, you will find detailed information on how to upgrade each component of your workspace and information on the necessary environments.

- Chapter 4, *Prerequisites for the Upgrade* [18] provides an overview of the software you need to install in order to build the new workspace.
- Chapter 5, *Changes in Supported Environments* [20] lists all the changes in the supported environments, so that you know, for example, if you need to install a new database.

- Chapter 6, *Removed Components and Features* [24] lists all major components, integrations or features which have been removed from the new CoreMedia version. This way, you can remove them from your workspace before you start the upgrade.
- Chapter 7, *Detailed Overview of Changes* [30] finally describes all the detailed changes in the new CoreMedia version per component. Here, you will learn if you have to adapt your customizations, configurations, deployment or other tasks.

4. Prerequisites for the Upgrade

First of all, in order to upgrade to *CoreMedia Content Cloud* v12, CoreMedia recommends to start from the latest *CoreMedia Content Cloud* v11 version.

The list below gives a detailed overview of the software you must have installed before the upgrade.

- Git >= 2.25.0

  The guide uses the Git command **restore** which is only available since Git version 2.25.0.

- NodeJS 20.x

  In order to compile the Studio Client workspace and all frontend related components, you need to install NodeJS in the latest version 20.x.

- pnpm 8.15.3

  In order to compile the Studio Client workspace and all frontend related components, you need to install pnpm as the package manager. Simply call `npm install -g pnpm@8.15.3` on the commandline.

- Configuring Access to CoreMedia npm Repository

  In *CoreMedia Content Cloud* v12 CoreMedia provides an npm registry to provide CoreMedia npm packages. To access this registry, you need a GitHub user token and need to set the credentials. The GitHub user needs to be a member of the coremedia-contributions organisation in GitHub. If in doubt, contact CoreMedia support to validate your permissions.

  1. Creating a GitHub Token

     Create a GitHub token as described in https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token. with the following parameters:

     - read:packages
     - read:org
     - read:user

  2. Configuring your npm Repositories

     a. Configure the npm repositories with the following pnpm calls:

```
pnpm config set @coremedia:registry https://npm.coremedia.io
pnpm config set @jangaroo:registry https://npm.coremedia.io
```

b.

```
pnpm login --registry=https://npm.coremedia.io
```

You will be asked for user and password. Enter your GitHub username (all lowercase) and the created GitHub token as the password.

5. Changes in Supported Environments

This chapter lists third-party software that is no longer supported by *CoreMedia Content Cloud* or which has been added.

5.1 Added Support

5.1.1 Java 17

CoreMedia 12 requires Java 17. Java 17 is the next Long Term Support version of the Java platform. See Section 7.12.1, “Java 17 Support” [62] for details.

5.1.2 NodeJS 20

NodeJS 20 (LTS) is now required to build the Frontend and Studio Client workspaces.

5.1.3 pnpm 8.15.3

Along with NodeJS 20, pnpm 8.15.3 is now required to build the Frontend and Studio Client workspaces.

5.1.4 Container Operating Systems

Ubuntu with OpenJDK 17 and Amazon Linux with Corretto 17 have been added as new container operating systems.

5.1.5 Container Environments

Containerd 1.6+ is the new supported container environment.

5.2 Removed Support

5.2.1 Databases

CoreMedia has removed support for the following database systems:

- Oracle 12c
- Amazon RDS for Oracle 12c
- Amazon RDS for PostgreSQL 13

5.2.2 Removed Container Environments

Support for the Containerd 1.4+ environment has been removed.

5.2.3 Removed Container Operating Systems

CoreMedia has removed support for the Debian and Amazon Linux with Corretto 11 operating systems.

5.2.4 MongoDB 4.0 and 4.4

CoreMedia has removed support for MongoDB 4.0 and 4.4. for Elastic Social and Collaboration and Workflows.

5.2.5 Removed Frontend Related Systems

CoreMedia has removed support for the following frontend related systems:

- NodeJS 16, 18
- pnpm 6.x, 7.1.5

- SenchaCMD
- Maven 3.8.4
- Java 11

6. Removed Components and Features

This section lists components and features which have been removed in *CoreMedia Content Cloud*. In the most cases, CoreMedia recommends that you remove the components from your workspace before upgrading to the new version.

6.1 Removal of the SiteManager

The SiteManager has been removed from CoreMedia CMS. The SiteManager was a tool to manage the structure of a website. It was used to create and manage pages, folders, and other content types.

6.2 Removed CKEditor 4 in Studio Rich-Text

Support for CKEditor 4 in Studio was already deprecated in CMCC 11 and has now been removed. If you are using CKEditor 4 refer to the CMCC 11 Upgrade Guide to learn how to migrate extensions to CKEditor 5.

6.3 Removed Support for JSPs

JSPs are no longer supported. Plain JSPs can still be used but JSP taglibs can no longer be loaded from Freemarker templates. Note that taglib URIs in JSPs and jinc files must be adjusted to the Jakarta Taglib URIs. http://java.sun.com/jsp/jstl/ must be replaced with jakarta.tags, for example. See https://jakarta.ee/specifications/tags/3.0/jakarta-tags-spec-3.0#multiple-tag-libraries and https://jakarta.ee/specifications/tags/3.0/tag-docs/index.html for further information

6.4 Removed OpenCalais Integration

The OpenCalais integration has been removed from the Blueprint.

6.5 Removed WebTrends

The WebTrends integration has been removed from the Blueprint.

7. Detailed Overview of Changes

This chapter provides an overview of the improvements in *CoreMedia Content Cloud* v12 over *CoreMedia Content Cloud* v11.

- Section 7.1, “Updated Third-Party Libraries” [31] lists updated third-party libraries.
- Section 7.2, “API Changes” [35] lists all API changes in tables and describes some other API changes of *CoreMedia Content Cloud* v12 in separate sections.
- Section 7.3, “Studio Client Changes” [38] describes changes in the Studio Client.
- Section 7.4, “Studio Server Changes” [43] describes changes in the Studio Server.
- Section 7.5, “Content Server Changes” [44] describes changes in the Content Servers.
- Section 7.6, “Workflow Server Changes” [45] describes changes in the Workflow Server.
- Section 7.7, “Content Application Engine Changes” [49] describes changes in the Content Application Engine.
- Section 7.8, “Feeder and Search Engine Changes” [50] describes changes in the Feeder and the Search Engine.
- Section 7.9, “Elastic Social Changes” [51] describes changes in the *CoreMedia Content Cloud* v12 elastic social features.
- Section 7.10, “Headless Server Changes” [52] describes changes in the *CoreMedia Content Cloud* v12 Headless Server.
- Section 7.11, “Commerce Integration Changes” [61] describes changes in the different CoreMedia commerce integrations.
- Section 7.12, “Blueprint Changes” [62] describes changes in the *CoreMedia Blueprint*.
- Section 7.13, “Deployment Changes” [70] describes changes in the deployment of the system.
- Section 7.14, “Frontend Workspace Changes” [73] describes changes in the Frontend workspace. This also includes frontend related topics in the *CoreMedia Blueprint*.
- Section 7.15, “Command Line Tool Changes” [76] describes changes in the command line tools of *CoreMedia Content Cloud*.

A complete list of changes is available on the https://documentation.coremedia.com/cm-cc-12/release-notes pages.

7.1 Updated Third-Party Libraries

This section lists updated third-party libraries in *CoreMedia Content Cloud* v12. You can find more API changes in the other change chapters.

7.1.1 Upgrade to Solr 9.4.1

Solr has been upgraded to release 9.4.1. See https://solr.apache.org/docs/9_4_1/changes/Changes.html for a list of changes. Alongside, a new coremedia/solr-base Docker image with tag 2.2-cm-9.4.1 was released.

As part of this change, the following transitive dependencies of Apache Solr have been updated to match versions used by Solr:

- Apache Zookeeper 3.9.1
- Eclipse Jetty 10.0.19
- FasterXML Jackson 2.16.1
- SLF4J 2.0.10 (CMCC 12 2401 only)

Since Solr 9.4.0, configuration of a proxy in communication of clients with Solr via HTTP/2 is possible. To enable a proxy, use these new properties:

- *solr.proxy-host*: Proxy host for Solr communication.
- *solr.proxy-port*: Proxy port for Solr communication.
- *solr.proxy-is-socks4*: SOCKS 4 flag for Solr proxy. Default is *false*.
- *solr.proxy-is-secure*: Secure flag for Solr proxy. Default is *false*.

Leaving *solr.proxy-host* or *solr.proxy-port* unset, disables proxy configuration and lets the clients communicate with Solr directly. Environment variables *HTTP_PROXY* and *HTTPS_PROXY* are not evaluated and thus have no effect.

7.1.2 Third-Party Update: Apache Tika and Transitive Dependencies

Apache Tika has been updated to version 3.0.0-BETA. As part of this change, the following transitive dependencies of Apache Tika have been updated to match versions used by Tika.

Updated dependencies:

- Apache Commons Codec 1.16.0
- Apache Commons Compress 1.25.0
- Apache Commons Lang3 3.14.0
- Apache Commons IO 2.13.0
- Apache Log4J API 2.21.1
- Apache PDFBox 3.0.1
- Apache POI 5.2.5
- Apache Tika 3.0.0-BETA
- Apache XMLBeans 5.2.0
- ASM 9.6
- Drew Noakes Metadata Extractor 2.19.0
- FasterXML Jackson 2.16.1
- Glassfish JAXB 4.0.4
- Jakarta Bind API 4.0.1
- Rome 2.1.0

If you use these libraries in project code, please check their respective release notes for changes and upgrade information.

(CMS-23952)

7.1.3 Spring Boot 3.2 and Related Third-Party Libraries

The following dependencies were upgraded to new major versions with notable changes:

- Spring Boot from 2.7 to 3.2. To upgrade and migrate custom sources, please see:
  - Spring Boot 3.0 Release Notes
  - Spring Boot 3.1 Release Notes
  - Spring Boot 3.2 Release Notes
- Spring Framework from 5.3 to 6.1. To upgrade and migrate custom sources, please see:
  - What's New in Spring Framework 6.x
  - Upgrading to Spring Framework 6.x
- Spring Security from 5.8 to 6.2. To upgrade and migrate custom sources, please see:
  - What’s New in Spring Security 6.0
  - What’s New in Spring Security 6.1
  - What’s New in Spring Security 6.2
  - Migrating to Spring Security 6.0
  - Migrating to Spring Security 6.2
- Spring Data BOM from 2021.2 to 2023.1. To upgrade and migrate custom sources, please see:
  - Spring Data 2022.0 (Turing) Release Notes

  - Spring Data 2023.0 (Ullman) Release Notes
  - Spring Data 2023.1 (Vaughan) Release Notes
- Spring Webflow from 2.5 to 3.0. The new major version is a compatibility release, allowing projects to migrate to the latest Spring versions without having to replace implementations of the abandoned Webflow project. See also Blog Post: Spring Web Flow 3.0.0 Released.
- Jakarta Activation API from 1.2 to 2.1. To upgrade and migrate custom sources, please note:
  - The dependency coordinates changed from `com.sun.activation:jakarta.activation` to `jakarta.activation:jakarta.activation-api`
  - The packages of the Java classes changed from `javax.activation` to `jakarta.activation`
- Jakarta Annotations API from 1.3 to 2.1. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.annotation` to `jakarta.annotation`
- Jakarta Dependency Injection API from 1 to 2.0 To upgrade and migrate custom sources, please note:
  - The dependency coordinates changed from `javax.inject:javax.inject` to `jakarta.inject:jakarta.inject-api`
  - The packages of the Java classes changed from `javax.inject` to `jakarta.inject`
- Jakarta Mail API from 1.6 to 2.1. To upgrade and migrate custom sources, please note:
  - The dependency coordinates changed from `com.sun.mail:jakarta.mail` to `jakarta.mail:jakarta.mail-api`
  - The packages of the Java classes changed from `javax.mail` to `jakarta.mail`
- Jakarta Servlet API from 4.0 to 6.0. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.servlet` to `jakarta.servlet`
- Jakarta Server Pages API from 2.3 to 3.1. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.servlet.jsp` to `jakarta.servlet.jsp`
- Jakarta Transactions API from 1.3 to 2.0. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.transaction` to `jakarta.transaction`
- Jakarta Validation API from 2.0 to 3.0. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.validation` to `jakarta.validation`

- Jakarta XML Binding API from 2.3 to 4.0. To upgrade and migrate custom sources, please note:
  - The packages of the Java classes changed from `javax.xml.bind` to `jakarta.xml.bind`
- JAXB Runtime from 2.3 to 4.0. To upgrade and migrate custom sources, please see Jakarta XML Binding - Release Notes.
- Hibernate Object/Relational Mapping from 5.6 to 6.4. To upgrade and migrate custom sources, please see Hibernate ORM Migration Guides.
- Hibernate Validator from 6.2 to 8.0. To upgrade and migrate custom sources, please see Hibernate Validator Migration Guide.
- SLF4J from 1.7 to 2.0. To upgrade and migrate custom sources, please see What has changed in SLF4J version 2.0.0?.
- Apache Tomcat from 9.0.82 to 10.1. To upgrade and migrate custom sources, please see Apache Tomcat Migration Guide.

7.1.4 Updated Third-Party Libraries

Apart of the updates that have been mentioned in the previous sections, many further libraries have been updated. Updating dependencies is a continuous task, each Core-Media release includes several updates to third-party libraries. As this upgrade guide is not tied to a particular CMCC 12 version, it does not make sense to list all minor or patch version updates here. Please refer to the CMCC 12 Release Notes for the CMCC 12 versions of interest.

7.2 API Changes

This section contains changes in the CoreMedia APIs. You will find more API changes in the component specific subsections.

7.2.1 Removal of conversionService Bean

The Unified API Spring Boot integration configured a bean `conversionService` of type `org.springframework.core.convert.support.DefaultConversionService` so that it was capable of converting strings to `ContentType` instances. This feature was previously used for the Blueprint Spring configuration classes `InterceptorsStudioAutoConfiguration`, `LcStudioLibComponentAutoConfiguration`, and `InferrersStudioConfiguration`]. The default conversion service bean has been removed, and the configuration classes have been adjusted to use explicit content type retrieval. Instead of

```
@Value("CMLinkable") ContentType contentType
```

the following code can be used to achieve the content type lookup

```
@Value("#{@contentRepository.getContentType('CMLinkable')}") ContentType
contentType
```

See https://docs.spring.io/spring-framework/reference/core/validation/convert.html#core-convert-Spring-config for further information on Spring type conversion.

This bean was removed from the Unified API Spring integration along with the capability to convert strings to content types automatically.

7.2.2 Removal of c.c.b.b.l.Optional

The Java class `com.coremedia.blueprint.base.links.Optional` was removed and replaced by `java.util.Optional` wrapping a string. The class was public API but only used by `com.coremedia.blueprint.base.links.LinkTargetPattern` to compute link targets.

7.2.3 Removal of c.c.o.w.MappedInterceptor

The Java class `com.coremedia.objectserver.web.MappedInterceptor` was introduced because of shortcomings of an early version of the Spring Framework class `org.springframework.web.servlet.handler.MappedInterceptor` (when CM 9 was released). The Java class `com.coremedia.objectserver.web.MappedInterceptor` was removed from CMCC 12, as the corresponding Spring Framework class no longer suffers from its initial shortcomings. Use `org.springframework.web.servlet.handler.MappedInterceptor` instead.

7.2.4 Blob API Changes

Recompile and Maybe Source Code Changes Necessary

The Blob API has been changed to support blobs larger than 2GB. This change affects the public API. You must at least recompile your project Blueprint workspace against the new CMCC version. Depending on your customizations, you may have to adjust some source code too, as explained below.

Supporting huge blobs implies that we cannot represent blob sizes as int values any longer, but had to switch to long. This required some changes that affect the public API.

To access a blob's size, a new UAPI method `com.coremedia.cap.common.Blob#getSizeLong()` has been introduced which returns the blob size as a long value. The still existing method `Blob#getSize():int has` been deprecated and will be removed in a future release. Please be aware that this deprecated method will fail with an exception when called for a blob of size 2GB and beyond since an int value cannot represent huge numbers (it's limited to `java.lang.Integer#MAX_VALUE)`. To be safe, only the new method Blob#getSizeLong()}}should be used in UAPI clients and Freemarker templates.

All usages of `Blob#getSize()` have been replaced by `Blob#getSizeLong()` in the Blueprint workspace. For backward compatibility, Blob#getSizeLong() is a default method that delegates to `Blob#getSize()`. Thus, existing custom implementations of the Blob interface will still work until we finally delete `Blob#getSize()`.

Furthermore, avoid the use of method `Blob#asBytes():bytes[]` if huge blobs are to be processed, as a Java array can only hold up to `java.lang.In` `teger#MAX_VALUE` bytes and thus cannot be used for huge blobs. Calling this method on a huge blob will result in an exception being thrown. Ideally, all blobs should be transferred via streams.

The interface `com.coremedia.cap.common.BlobService` features various blob creation methods which expect a size argument. The type of the size argument has been changed from int to long for all those methods.

In the interface `com.coremedia.cap.common.CapConnectionManager` the methods `setMaxCachedBlobSize` and `getMaxCachedBlobSize` have been changed from int to long.

7.3 Studio Client Changes

This section describes the changes in the user interface of CoreMedia Studio and in the build of Studio client.

7.3.1 Removal of CKEditor 4 Support

Starting with CMCC 12.2401.1 CKEditor 4 is not supported anymore. Corresponding references in the Studio Developer Manual have been removed.

Note, that in upcoming releases, left-over traces in CoreMedia Blueprint will be removed.

If you are updating from an adapted CKEditor 4 integration, please see CMCC 11 documentation for upgrade hints. In general the update can be summarized with the following steps:

For any custom plugin you wrote, take its specification and rewrite it based on the CKEditor 5 architecture. You will benefit from a better architecture and more flexibility when it comes to integrating native CKEditor 5 plugins for use in CoreMedia Studio.

7.3.2 Sencha Cmd Removal

Sencha CMD is no longer needed to build the studio-client. This means Java 11 and Sencha CMD no longer need to be installed. Because this is one less step during the studio-client build the build is now also faster.

For this some (third-party) packages have been updated to new (major) versions. Make sure to do the same as well for every additional package in the studio-client workspace:

- @coremedia/sencha-* to “^1.1.0”
- @jangaroo/ext-ts to “^2.0.0“
- remaining @jangaroo/* to “^3.0.0“ (except for @jangaroo/eslint-config, see Section 7.3.3, “Changed Linter Rules” [39])
- typescript to “^5.3.3”
- eslint to “^8.56.0”
- rimraf to “^5.0.5”

The Dockerfile in `apps/studio-client/Dockerfile.tooling` no longer has a `sencha-layer` and the corresponding installation for Java 11 and Sencha

CMD have been removed. Please also mind the dependency adjustments in the `npm-tooling-base` layer.

7.3.3 Changed Linter Rules

The linter rules in all studio-client packages have been updated to a more common rule set. In addition to this the rule set is no longer part of the Jangaroo tooling but of the studio-client workspace.

While it was useful (and even required) for the TypeScript migration that the Jangaroo tooling has brought the corresponding linter rules to result in a mergeable code base, it is now more focused on the actual build process and the resulting runtime environment. This is why the package `@jangaroo/eslint-config` is no longer available from v3 on.

There are now two rule sets in the studio-client workspace: `@coremedia/eslint-config-studio-client` and `@coremedia/eslint-config-studio-client-ext`. The former has strict rules for ExtJS independent TypeScript code while the latter is meant for packages containing ExtJS components which require some exceptions to the strict rules.

All TypeScript code in the studio-client workspace has been updated to apply to the new rule sets.

Upgrade steps:

Before merging the 2404.1 code base make sure to be on 2307.3 / 2401.3 or later. It is assumed that the studio-client workspace builds without any errors.

To prepare the merge, add the following pnpm overrides to your `apps/studio-client/package.json`. Make sure that the version matches the version of the studio-client workspace you are merging into. For 2404.1 it would be:

```
Linter overrides:
  "pnpm": {
    "overrides": {
      "@jangaroo/eslint-config":
"npm:@coremedia/eslint-config-studio-client-ext@2404.1.0",

"@coremedia-blueprint/studio-client.blueprint-doctypes>@jangaroo/eslint-config":
 "npm:@coremedia/eslint-config-studio-client@2404.1.0",

"@coremedia-blueprint/studio-client.ecommerce-doctypes>@jangaroo/eslint-config":
 "npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.alx>@jangaroo/eslint-config":
"npm:@coremedia/eslint-config-studio-client@2404.1.0",
     "@coremedia-blueprint/studio-client.alx-google>@jangaroo/eslint-config":
 "npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.am>@jangaroo/eslint-config":
"npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.catalog>@jangaroo/eslint-config":
 "npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.es>@jangaroo/eslint-config":
```

```
"npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.lc>@jangaroo/eslint-config":
"npm:@coremedia/eslint-config-studio-client@2404.1.0",
      "@coremedia-blueprint/studio-client.p13n>@jangaroo/eslint-config":
"npm:@coremedia/eslint-config-studio-client@2404.1.0"
    }
  }
```

This temporarily replaces the Jangaroo linter rules in all packages with the new studio-client rules coming with the 2404.1 release. While the ExtJS specific rule set is the default, all ExtJS independent packages get the stricter rule set. You do not need to complement this list with your own packages in this step as new packages will not cause any merge conflicts.

In the folder `apps/studio-client` run `pnpm install` to apply the overrides.

After the installation was successful, run `pnpm -r --no-bail run lint` in the same folder to fix all automatically fixable linter errors.

Revert the changes made to the `apps/studio-client/package.json` and also restore the `apps/studio-client/pnpm-lock.yaml`.

Check in all code changes.

Ignore-merge the original, but already reformatted state of the version you are coming from. You can find a corresponding tag with the suffix "-converted" in the CoreMedia Blueprint Workspace. E.g. if you come from 2401.3 there is a tag `cmcc-12-2401.3-converted` which can be ignore-merged using the following command: `git merge -s ours cmcc-12-2401.3-converted`

Merge the target version (e.g. cmcc-12-2404.1) into your vendor branch.

Perform all third-party upgrades as described in the Section 7.3.2, "Sencha Cmd Removal" [38] section.

Replace the "devDependency" `@jangaroo/eslint-config` with the corresponding `@coremedia/eslint-config-studio-client` or `@coremedia/eslint-config-studio-client-ext` configuration in the `package.json` of all your additional packages.

Remove the "--fix" option from the lint script in the `package.json` files of all your additional packages.

Adjust the .eslintrc.js files (if present) in all packages accordingly to match the utilized rule set.

Run `pnpm install` in all packages to apply the changes and check in the updated `apps/studio-client/pnpm-lock.yaml`.

Run `pnpm -r --no-bail run lint --fix` (mind the newly added "--fix") in all packages to fix all automatically fixable linter errors.

(optionally) Check the linting results of the previous run and fix the remaining errors manually that could not be automatically fixed by the linter.

7.3.4 Stricter CSP Rules

The Content Security Rules for Studio Client have been hardened to prevent attackers from uploading JavaScript code into CMS Content and exploit possible XSS vulnerabilities to execute that code in the context of another Studio user. If you have added any custom scripts to Studio Client which are *not* deployed under the paths `/packages/` or `/resources/`, they will now be blocked by the stricter CSP rules, which is reported in the browser console.

To fix this, you need to allow-list those scripts in `apps/studio-client/apps/main/base-app/sencha/resources/config-init.js` by adding their paths to the allow-list `[..., "resources/", "packages/"]`.

7.3.5 StatefulDateTimeField's Configuration has Changed

The date and time formats of the component `StatefulDateTimeField` are now configured in `BaseModels_properties`. The properties `Date_format` and `Time_format` in `UI_properties` are now removed.

7.3.6 Changed icons in Studio to SVG format

All usages of the *CoreIcon_properties* have been removed from the workspace. The utility method *SvgIconUtil#getIconStyleClassForSvgIcon* is used instead. This method can either be used directly inside a component or inside a resource bundle class.

Additionally, the following resource bundle changes have been made:

- The deprecated resource bundle *StandardDocumentTypes_properties* has been removed. The resource bundle *ContentTypes_properties* must be used instead.
- The module *core-icons* which included the *CoreIcons_properties.ts* has been removed.
- The module *collaboration-icons* which included the *CollaborationIcons_properties.ts* has been removed.
- Job icons are now read from the new properties class *Job_properties*. If your are using custom icons for jobs, use the *CopyResourceBundleProperties* to apply your icon definitions to this resource bundle.

- Content Hub icons are now read from the new properties class *ContentHub_properties*. If your are using custom icons for a Content Hub adapter, use the *CopyResource-BundleProperties* to apply your icon definitions to this resource bundle.

7.3.7 New Property for UploadSettings

No Changes Necessary

When you are happy with the old blob size limit, you do not need to change anything.

Since CoreMedia CMCC 12 supports now blobs larger than 2GB, the `UploadSettings` settings have a new property `maxFileSizeMB` to allow for the configuration of upload limits beyond 2GB. When present, this setting overrides the old value maxFileSize. The default value of 64MB is unchanged.

7.4 Studio Server Changes

This section describes changes applied to the Studio server.

7.4.1 Removed Property for Spring Boot Maven plugin

The `additionalProfiles` Maven property that could have been used to set profiles to activate when starting the Studio Server Spring Boot app locally via the Spring Boot Maven plugin was removed. When additional profiles need to be set, use the `spring-boot.run.profiles` property of the plugin. This will override the plugin configuration. Therefore, make sure to also activate the dev and local profiles if needed. If previously the command `mvn spring-boot:run -DadditionalProfiles=my-profile` was used, now use the equivalent `mvn spring-boot:run -Dspring-boot.run.profiles=dev,local,my-profile`. Alternatively, CoreMedia apps can still be started conveniently from within the IDE. Consult the documentation for further details.

7.5 Content Server Changes

This section describes changes applied to the different content servers.

7.5.1 Database Schema Changes

Necessary Work

You should do this test in order to minimize impact on your live environment.

To represent huge blob sizes, the database schemas have been adjusted. Upon first startup, database schemas will be migrated automatically. It is recommended to record migration times during a test run on a copy of productive data to estimate downtimes during production roll-out.

7.6 Workflow Server Changes

This section describes changes applied to the Workflow Server and the deployed workflows.

7.6.1 Removed Deprecated Workflow Java API

The following deprecated classes and methods have been removed from the Workflow Server API and cannot be used in custom code or workflow definitions anymore:

- Class `com.coremedia.workflow.common.util.AbstractExternalManager`
- Class `com.coremedia.workflow.common.util.LongActionRegistry`
- Class `com.coremedia.workflow.common.util.RegisteringLongAction`
- Class `com.coremedia.workflow.common.util.SpringCollectiveLongAction`
- Class `com.coremedia.workflow.common.util.SpringDelegateAction`
- Class `com.coremedia.workflow.common.util.SpringDelegateLongAction`
- Method `com.coremedia.workflow.WfServer#hasManager`
- Method `com.coremedia.workflow.WfServer#getManager`
- Method `com.coremedia.workflow.common.actions.Log#setFatal`
- Method `com.coremedia.workflow.common.util.WithSpringContext#getBean`
- Method `com.coremedia.workflow.common.util.SpringAwareAction#getBean`
- Method `com.coremedia.workflow.common.util.SpringAwareExpression#getBean`
- Method `com.coremedia.workflow.common.util.SpringAwareLongAction#getBean`

Hints on how to replace these classes and methods are documented as deprecation note in the Javadoc of CMCC 11. In general, it's recommended to use Unified API interfaces to implement custom managers, and to use SpringAware*Action classes to implement actions that interact with the Spring context.

If you have used some of these classes in custom workflow definitions, then you should make sure to finish all running workflow instances that were started from these definitions before upgrading. Old workflow definitions that were using these classes will not be usable anymore, and may have to be destroyed using `cm workflowconverter -x`.

7.6.2 Replacement of Deprecated Workflow Server Log Facility

The location of loggers for certain classes changed. In case you are interested in those logs and for example applied custom logging configuration, adapt it to the following changes and decide more precisely which loggers are of importance for you. The following classes are now using a logger in their own class instead of the logger of the `com.coremedia.workflow.impl.server.Server` class:

- com.coremedia.cotopaxi.wfserver.ServerRightsPolicyMarshallerResolver
- classes extending com.coremedia.workflow.impl.server.AbstractProcessSignalHandler
- classes extending com.coremedia.workflow.impl.server.AbstractTaskSignalHandler
- com.coremedia.workflow.impl.server.Condition
- com.coremedia.workflow.impl.server.ContentManager
- com.coremedia.workflow.impl.server.ErrorLog
- com.coremedia.workflow.impl.server.InternalSession
- com.coremedia.workflow.impl.server.LightweightSession
- com.coremedia.workflow.impl.server.MasterSession
- com.coremedia.workflow.impl.server.ObjectRepositoryImpl
- com.coremedia.workflow.impl.server.ParameterAssignment
- com.coremedia.workflow.impl.server.ProcessValidator
- com.coremedia.workflow.impl.server.SessionManager
- com.coremedia.workflow.impl.server.TaskImpl
- com.coremedia.workflow.impl.server.TimerManager
- com.coremedia.workflow.impl.server.TimerMap
- com.coremedia.workflow.impl.server.TransactionManager
- com.coremedia.workflow.impl.server.Transformer
- com.coremedia.workflow.adapters.persistence.GenericRDBMSAdapter
- com.coremedia.workflow.beanparser.ProcessDefinitionFactory
- com.coremedia.workflow.handlers.RunActionTimerHandler
- com.coremedia.workflow.handlers.SkipUserTaskTimerHandler
- com.coremedia.workflow.servlets.CommandServlet
- com.coremedia.workflow.servlets.IORServlet
- com.coremedia.workflow.servlets.StatsServlet
- com.coremedia.workflow.tasks.ForkSubprocessRunTaskHandler
- com.coremedia.workflow.validation.OfflineProcessDefinitionFactory

7.6.3 Deprecated ArchiveProcess Workflow Action: Functionality Removed

NOTE

When you use custom workflow definitions with the ArchiveProcess action, you have to adapt your workflow definitions.

The `ArchiveProcess` action in workflow definitions was used to store a finished workflow in the database. It was replaced by the `ArchiveProcessFinalAction` in CMCC 11, because it can also archive data from aborted workflows, while the `ArchiveProcess` action was only capable to archive data from workflows that completed successfully.

In your custom workflow definitions, replace <Action> tags which contain the *`class`* attribute `ArchiveProcess` with the <FinalAction> tag with the *`class`* attribute `ArchiveProcessFinalAction`.

The `ArchiveProcess` class still exists for backwards-compatibility of old serialized workflow definitions, but the action has no functionality anymore. Only workflows with configured `ArchiveProcessFinalAction` will appear in the workflow archive.

7.6.4 Workflow Server Database Tables

The *Workflow Server* will automatically create new tables in the 'cm_management' database:

- `WfArchivedProcessInstances`
- `WfArchivedTaskInstances`
- `WfArchivedVariables`

These names are reserved and cannot be used as document type names.

7.6.5 Property 'usecaplist' not Used by Workflow Server anymore

In the Workflow-Server previously, the `usecaplist` property was used to decide whether to connect to the CapListRepository. Now, the `repository.caplist.connect` is used for this. Exchange any usages you might have of the old `usecaplist` property to connect the CapListRepository in the Workflow Server for the new `repository.caplist.connect` property.

7.6.6 Workflow Archiving Properties

Configure the process archiving in the in-memory setup with the following properties also documented in the Deployment Dependencies section:

- workflow.archive.endpoint-url instead of studio.url
- workflow.archive.username instead of studio.user
- workflow.archive.password instead of studio.password

7.7 Content Application Engine Changes

This section describes changes applied to the Content Application Engine (CAE).

7.7.1 Removal of Artifact com.coremedia.cms:cae-util

The CAE core artifact `com.coremedia.cms:cae-util` was removed. The beans provided by resource `com/coremedia/cae/uapi-services.xml` or rather `com.coremedia.objectserver.util.config.CaeUtilConfiguration` as the XML file was already deprecated, of the cae-util dependency are now provided by the Spring configuration class `com.coremedia.objectserver.config.ContentBeanServicesConfiguration` of `com.coremedia.cms:cae-contentbeanservices`. If the beans from the `cae-util` dependency are required, migrating to the new providing configuration class is as well.

7.7.2 ccbbli.RuleUrlPrefixResolver has new Constructor

The class `com.coremedia.blueprint.base.links.impl.RuleUrlPrefixResolver` has a new `RuleUrlPrefixResolver(List<RuleProvider>)` constructor. The `PostConstruct` method was removed.

7.8 Feeder and Search Engine Changes

This chapter contains detailed information about necessary upgrade steps for search components.

7.8.1 Known Issue

Spring Boot 3.2 supports Jetty 12 but the most recent SolrJ version (9.4.1) still depends on Jetty 10 as HTTP client implementation. Hence, CMCC 12 apps connecting to Solr depend on Jetty 10 HTTP client. This incompatibility may lead to Exceptions caused by java.lang.NoSuchMethodError: 'org.eclipse.jetty.client.Request org.eclipse.jetty.client.HttpClient.newRequest(java.net.URI)' when an application depends on SolrJ and uses a Spring RestTemplate without explicitly configuring a HTTP client request factory. Solutions are to either adding a runtime dependency on `org.apache.httpcomponents.client5:httpclient5` (which takes precedence over the Jetty HTTP client) or to explicitly configure the HTTP client request factory.

7.9 Elastic Social Changes

This section describes changes applied to the Elastic Social integration.

7.9.1 Elastic Social Moderation With CKEditor 5

Probably no Changes Necessary

Unless you adapted BBCode handling within the various layers (see next section below), no upgrade steps are required. If you adapted any of the layers listed below, please consult our Support Team. They will provide detailed information on the various aspects.

Elastic Social moderation in CoreMedia Studio has been updated to use CKEditor 5 instead of CKEditor 4.

This change especially includes new CKEditor 5 editor types, that support data backed by BBCode as well as support to managing blocked words to provide hints to editors where comments/review posts contain problematic words or character sequences.

BBCode Support Layers

For alignment of BBCode processing within CoreMedia Content Cloud, the following relevant layers have been adapted:

- Elastic Social Moderation backed by CKEditor (now CKEditor 5)
- CAE Delivery backed by KefirBB (with mapping configuration options backed by `kefirbb.xml`)
- Contribution to Article backed by `BbCodeToCoreMediaRichtextTransformer`, that ships with CoreMedia Blueprint.

The BBCode language features officially supported are described in the Elastic Social manual (search for BBCode). The corresponding section has been updated to reflect enhancements like quoted arguments support for the `[url]` BBCode tag.

7.10 Headless Server Changes

This chapter contains detailed information about necessary upgrade steps for the Headless Server.

7.10.1 Third-Party Library Changes

Only Changes Necessary when Libraries were Used

Removals

The following libraries were removed, because the corresponding projects are not updated or supported anymore. Therefore, you cannot use their APIs anymore.

- `com.graphql-java:graphql-java-spring-webmvc`
- `com.graphql-java-kickstart:graphql-spring-boot-starter`

If you have used some of the classes provided by these libraries in your custom code, this code will not compile anymore. Depending on what your code was doing, it might be necessary to reimplement it, using the API and principles of the newly introduced Spring-GraphQL library.

Together with the removal of the mentioned libraries, the following classes were removed and reimplemented for the new Spring-GraphQL library as private API.

- `com.coremedia.caas.web.wiring.GraphQLInvocationImpl`
- `com.coremedia.caas.web.view.ViewValidator`
- `com.coremedia.caas.web.persistedqueries.PersistedQueryExecutor`
- `com.coremedia.caas.web.persistedqueries.DefaultPersistedQueriesLoader`
- `com.coremedia.caas.web.persistedqueries.DefaultQueryNormalizer`
- `com.coremedia.caas.web.persistedqueries.Hash`

Note, that some private API classes,for example, the controller for the /graphql endpoint, were also removed.

Additions

As a replacement for the removed libraries, Headless Server now uses the new Spring-GraphQL library `org.springframework.graphql:spring-graphql:1.2.5`.

Some of the mentioned removed classes were reimplemented based on the new Spring-GraphQL library. For further reference see the official documentation of Spring-GraphQL at https://docs.spring.io/spring-graphql/reference/index.html.

Updates

The following libraries were updated:

- `com.graphql-java:graphql-java:20.7`
- `com.graphql-java:java-dataloader:3.2.2`

If you have based custom code on some of the classes provided by these libraries, they might not compile anymore as there were some breaking changes introduced. For details please see the original release note at https://github.com/graphql-java/graphql-java/release and https://github.com/graphql-java/java-dataloader/releases .

Also note, that Headless Server is currently not using the latest version of GraphQL-Java. The version used is the required version of the Spring-GraphQL library.

7.10.2 Configuration Properties Changes

Probably Necessary to Check and Change

Due to the newly used Spring-GraphQL library, some configuration properties have been renamed and have sometimes new default values. Check all the properties.

Schema Introspection

The Spring-GraphQL library comes with its own implementation of schema introspection and with it also its own configuration property. Thus, the configuration property was removed in favor for Spring-GraphQLs configuration property.

Removed:	caas.graphql.introspectionEnabled
Replaced by:	spring.graphql.schema.introspection.enabled

Type:	Boolean
Default:	false

REST Mapping Controller

The REST mapping for persisted queries was previously handled by a controller, which was removed and replaced by a Spring-GraphQL-Handler and an autoconfiguration class.

Removed:	caas.graphql-restmapping-controller.enabled

To disable the REST mapping feature, add the autoconfiguration class to the SpringBoot exclude list, for example, in an environment variable.

```
SPRING_AUTOCONFIGURE_EXCLUDE=com.coremedia.caas.web.rest.RestMappingAutoConfiguration
```

GraphQL Endpoint

The configuration of the GraphQL endpoint is now handled by the Spring-GraphQL library. If you have not changed the standard value `/graphql`, no further action is necessary.

Removed:	graphql.url
Replaced by:	spring.graphql.path
Type:	String
Default:	/graphql

GraphiQL Endpoint

The web interface GraphiQL (pronounced 'graphicle') to the /graphql endpoint is now part of Spring-GraphQL, thus configuration is also handled by Spring-GraphQL.

Removed:	graphiql.enabled
Replaced by:	spring.graphql.graphiql.enabled
Type:	Boolean
Default:	false

Deployment behind Reverse-Proxy Server

When Headless Server is deployed behind a reverse proxy service and the outside protocol is HTTPS, Spring should be configured to forward certain HTTP headers. Otherwise, the protocol, for example, on the Swagger page will cause "Mixed Content" JavaScript issues, because the page tries to issue non-secure request while the page itself is SSL.

To configure this behavior, either use an environment variable or a configuration property.

Environment variable	SERVER_FORWARD_HEADERS_STRATEGY: "framework"
Configuration property	server.forward-headers-strategy=framework

7.10.3 Schema Loading API Change

The GraphQL schema loading used to work by path patterns. To be more flexible regarding the ability to enable or disable a feature which may come with a supplementary schema fragment, all schema fragments delivered out of the box now come as regular Spring Beans with the annotation `@Qualifier(“graphqlSchemaResource”)`.

All Spring Beans of the type `org.springframework.core.io.Resource` with that qualifier are automatically collected and added to the resulting schema.

Deployments consisting of custom extensions with a schema fragment must update their implementation to detect the schema extension.

```
Example Code Snippet:

@Bean
@Qualifier("graphqlSchemaResource")
public Resource mySchemaResource() throws IOException {
  PathMatchingResourcePatternResolver loader = new
PathMatchingResourcePatternResolver();
  return
Arrays.stream(loader.getResources("classpath*:graphql/my-path/my-schema-extension.graphql"))

          .findFirst()
          .orElseThrow(() -> new IOException("GraphQl schema resource
graphql/my-path/my-schema-extension.graphql' not found."));
}
```

Note, that the pattern-based detection of GraphQL schema files also still works. The Spring-GraphQL library detects by default all schema files on the class path with the file type-extension `.graphqls`.

7.10.4 Query Root Extension

If you extend not only existing GraphQL types from the content schema but also the query root type itself, the binding between the HTTP transport tier and the schema must be implemented by a Spring controller class containing a method named like the new query root property. The method should deliver your original root object.

Example:

```
# GraphQL-Schema extension:
    extend type Query {
```

```
    customRoot: CustomRoot
    }

    type CustomRoot {
    types: [CustomType]
    }

    type CustomType {
    name: String
    }

    ------------------------

    // Java class
    @Controller
    public class CustomRootController {

    private final CustomRoot customRoot;

    public CustomRootController (CustomRoot customRoot) {
    this.customRoot = customRoot;
    }

    @QueryMapping
    public CustomRoot customProperty() {
    return customRoot;
    }
    }
```

7.10.5 Controller for Persisted Queries Removed

Due to the introduction of Spring-GraphQL, the controller responsible for handling of persisted queries and related classes was removed. All configuration properties remain unchanged, and the behavior of all persisted query related handling remains as it was.

To disable the persisted query feature, add the autoconfiguration class to SpringBoots exclude list, for example, in an environment variable.

```
SPRING_AUTOCONFIGURE_EXCLUDE=
com.coremedia.caas.web.persistedqueries.impl.PersistedQueryAutoConfiguration
```

The following interfaces remained public API:

- `com.coremedia.caas.web.persistedqueries.PersistedQueriesLoader`
- `com.coremedia.caas.web.persistedqueries.QueryNormalizer`

Headless Server delivers default implementation for both interfaces. The Spring Beans are created with the annotation `@ConditionalOnMissingBean`. This means that you can easily replace the default implementation by your own implementation simply by creating a Spring Bean with that interface.

Due to the fact, that persisted queries are not handled via its own controller but instead internally of the GraphQL framework, the HTTP response codes have changed in behavior. Formerly, the controller responded with HTTP-FORBIDDEN/403 or HTTP-NOT-FOUND/404, if a persisted query was not found or it was tried to send a query while only persisted queries were allowed.

Headless-Server now always responds with HTTP-OK/200 and an error message in the responded JSON. This is also true, if the allow list is activated. [see Section 7.10.6, "Allow List" [57]].

7.10.6 Allow List

Changed property name

Formerly known under the name whitelist, the feature to allow persisted queries only was renamed to allow list. Thus, the configuration property was renamed correspondingly:

Removed:	caas.persisted-queries.whitelist
Replaced by:	caas.persisted-queries.allow-list
Type:	Boolean
Default:	false

7.10.7 PreparsedDocumentProvider

The execution of persisted queries is now handled as part of a so called `PreparsedDocumentProvider`. `PreparsedDocumentProvider` is a Java interface provided by the GraphQL Java library.

Earlier versions of Headless Server created an instance of `PreparsedDocumentProvider` within `CaasConfig`. That instance was removed from Blueprint.

Headless Server comes with its own implementation of `PreparsedDocumentProvider`, which is still handling basic query pre-parsing as well as Apollo persisted queries and server-side named persisted queries.

The `PreparsedDocumentProvider` is automatically instantiated by a Spring Boot autoconfiguration with the `@ConditionalOnMissingBean` annotation. Thus, customers may replace the CoreMedia default implementation simply by providing their own implementation of `PreparsedDocumentProvider` as a regular Spring bean.

If you already use a custom implementation of a `PreparsedDocumentProvider`, you will overwrite the provided default, thus effectively disable the execution of persisted queries in Headless Server, although the invocation on the GraphQL handler will remain intact.

7.10.8 REST Access and Persisted Query Mapping

The following classes regarding REST mapping were revoked from the public API:

- `com.coremedia.caas.web.GraphQLRestMappingConfig`
- `com.coremedia.caas.web.GraphQLRestMappingConfigurationProperties`

All configuration properties but `caas.graphql-restmapping-controller.enabled` remain unchanged (see section "REST Mapping Controller" [54]).

7.10.9 JSLT Transformation of Persisted Query Responses

Adapt JSLT Transformation Templates

All JSLT transformation templates must be adapted. They do not work anymore on the `data` property but one level deeper. Effectively, this means, that in all JSLT templates all paths starting with `.data` must be changed simply by removing '.data' from the transformation path.

Due to the invocation within the GraphQL context of Spring GraphQL, the JSLT transformation changed its behavior slightly.

Formerly, the JSLT transformation worked on the complete JSON graph starting with the `data` property of any GraphQL response. Now, the JSLT transformation works directly on the GraphQL response, meaning one level deeper. Therefore, the JSLT transformation templates had to be adapted to result in the same JSON response.

7.10.10 CORS

Changed Behavior when no CORS Configuration is Used

The default behavior on a missing configuration however was changed correspondingly to the CORS configuration provided by Spring GraphQL. On missing configuration, CORS protection is now disabled. Consequently, this change is considered breaking for those deployments, missing any CORS configuration. Existing CORS configurations should work as before.

With the introduction of the Spring GraphQL library, CORS got its own configuration properties for the `/graphql` endpoint. When the configuration properties are missing, CORS is effectively disabled for the `/graphql endpoint`, resulting in the so called Same-Origin-Policy. This means, CORS preflight requests are denied and CORS protection on the endpoint is disabled. The Same-Origin-Policy is effectively enforced by the client browser.

The generic CORS configuration properties `caas.cors.*` are still available, as they are potentially necessary for a CORS configuration on other endpoints, like `/caas/v1/media`.

7.10.11 Deprecated getSize() Method

Only Necessary when Large Blobs are Used

At the moment, you do not have to do anything, if you do not use large blobs. However, since the old method getSize() is deprecated, it is recommended to use the new method getSizeLong() instead.

The Headless Server API class `com.coremedia.caas.model.adapter.ContentBlobAdapter` also has a method `getSize()`, which has been deprecated in favor of a new method `getSizeLong()`, just like the UAPI `Blob` interface. This change has also been done in the GraphQL schema where a new field sizeLong was added to the `Blob` interface. The old field size is now deprecated.

7.10.12 Headless Server Uses Auto Configured Task Scheduler Bean

Headless Server created its own executor service which was configurable via `caas.rest.num-threads`. It now uses the auto configured task execution service. The property `caas.rest.num-threads` was removed. Use `spring.task.execution.pool.max-size` instead.

7.10.13 Replace Springfox with Springdoc

Springfox was replaced by Springdoc. This change is considered breaking only for customers using custom Spring controllers. If that's the case, the replacement of annotations might be required as described in the Springdoc migration guide: https://springdoc.org/v2/#migrating-from-springfox. The swagger documentation is still accessed the same way as before.

7.10.14 Media hash calculation Changed

Previously the media hash calculation for the media endpoint included all properties from site or global image transformation settings. This led to an unexpected change of the media hash, when changing non-relevant transformation properties, for example, `minWidth`. This change introduces the application property `caas.media.hash-property-names`, which defaults to the list of these values: `"width", "height", "widthRatio", "heightRatio"`.

This change is considered breaking, because all media URLs will change regarding the media hash. The expected impact is not high because wrong hashes issue a HTTP redirect (301) to the correct URL.

7.11 Commerce Integration Changes

This section describes changes applied to the different commerce integration components.

7.11.1 Commerce Adapter Updates

Changes only necessary with custom adapter

The versions of the Commerce Client API and the Commerce Adapters for the Commerce Hub have been updated from 2.2 to 3.0 versions.

There haven't been any changes to the gRPC protocol, hence the 2.2 and 3.0 adapters can be used with CMCC 11 and CMCC 12. For more detailed information, please refer to the release notes of the respective adapter:

- Commerce Client API and Mock Adapter for Commerce Hub
- Adapter for SAP Commerce Cloud
- Adapter for Salesforce Commerce Cloud B2C
- Adapter for HCL Commerce

The base adapter configuration used to pre-configure the readiness health group by setting `management.endpoint.health.group.readiness.include=readinessState,catalogRepository,storeRepository` has been removed. This proved to cause trouble when an adapter implementation disabled one of the health indicators of that list. Hence, the readiness group configuration was removed from the base adapter and now has to be configured on commerce adapter vendor implementations.

7.12 Blueprint Changes

This section describes changes in the CoreMedia Blueprint workspace.

7.12.1 Java 17 Support

Adapt your environments to use Java 17

CoreMedia 12 requires Java 17. Java 17 is the next Long Term Support version of the Java platform. All components have been updated to work with Java 17. Java 17 is now also required for building the workspace and running the applications, Java 11 will not work anymore.

Since *Sencha Cmd* The exception is the *studio-client* workspace which uses *Sencha Cmd* to build, which still requires Java 11. From CMCC 2404.1 on, CMCC 12 will not require *Sencha Cmd* anymore. Please refer to section Section 7.3.2, “Sencha Cmd Removal” [38] for further details.

Update your development and production environments to use Java 17.

Also note that a new Java 17 Base Image is provided that must be used for building Docker images. Refer to https://github.com/coremedia-contributions/docker.java-application-base/releases/ for the corresponding 2.1 release and changes associated with it.

If you are using artifact *cglib:cglib* in your own project code, you may want to consider switching to Spring's shadowed version of *cglib* provided by *org.springframework:spring-core*. *cglib* is not maintained actively any longer and presumably not compatible with Java 17, see https://github.com/cglib/cglib. The CoreMedia Blueprint workspace has been migrated to Spring's shadowed version of *cglib* and, consequently, does not define a *cglib*artifact version any longer. Should you want to migrate use of *cglib* in your own code, you need to replace package *net.sf.cglib* with *org.springframework.cglib*.

For handling of GIFs in the built-in image transformation we make use of the package *java.desktop/com.sun.imageio.plugins.gif* which is not exported and thus not usable with Java 17 by default. To allow access the following JVM argument must be added:

```
--add-exports java.desktop/com.sun.imageio.plugins.gif=ALL-UNNAMED
```

This argument is already added to the docker images, *spring-boot-maven-plugin* configuration and the provided IDEA run configuration files of the apps that perform image transformation (*studio-server, cae, headless-server*). If you have a custom deployment process or use GIF transformation in other applications, be aware to add this export. The Java 17 Base Image is prepared to add other *add-exports* or *add-opens* arguments if you need to via environment variables *JAVA_ADD_EXPORTS* and *JAVA_ADD_OPENS*.

7.12.2 Update to Spring Boot 3.2

Check your workspace

When you are still using `spring.factories` for registering Spring Boot auto-configuration or the `@Required` annotation, you have to adapt your code.

CoreMedia CMCC 12 depends on Spring Boot 3.2 (https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-3.2-Release-Notes) which in turn depends on Spring Framework 6 (https://github.com/spring-projects/spring-framework/wiki/What's-New-in-Spring-Framework-6.x).

Spring Framework 6 uses Java 17 on the source level and so does CoreMedia 12. Spring Framework 6 requires Jakarta EE 9 as minimum version. Jakarta EE dependencies have been updated in CoreMedia 12 to meet this requirement. Consequently, most of the existing javax.* imports have been replaced with jakarta.* imports. Prominent examples of such imports which are highly likely to be present in most projects are classes of the javax.servlet package which must be replaced with jakarta.servlet.

Spring Framework 6 changed the injection point resolution implementation so that the compiler flag *`-parameters`* is now mandatory, see https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-6.x#parameter-name-retention for further details. Add the compiler option *`<maven.compiler.parameters>true</maven.compiler.parameters>`* to project-specific instances of the maven-compiler-plugin and update the project-specific instances of the plugin to version 3.9.0.

The Spring Boot Maven Plugin 3 no longer supports disabling forking. It now always forks a JVM and needs to configure it using the *`spring-boot.run.jvmArguments`* command line property. See https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/#run for further details.

Spring Web MVC controller detection changed with Spring Framework 6. Spring Web MVC no longer detects controllers based solely on a type-level @RequestMapping annotation. Use the @Controller annotation instead. See https://github.com/spring-projects/spring-framework/wiki/Upgrading-to-Spring-Framework-6.x#web-applications-1 for further details.

Spring Framework 6 changed configuration class loading. Apps which import a configuration class in another (auto-)configuration class might run into a BeanDefinitionOverrideException if the imported configuration class is configured as an ordinary Spring bean in an imported Spring XML configuration file as well. To overcome this, such bean definition occurrences in Spring XML configuration files can be replaced by component scans. The recommendation is to use a regular expression pattern for the component scan, which narrows down the results to exactly the configuration class which should be replaced by it.

Spring MVC 6 deprecated trailing slash matching and changed the default to false. Therefore, URLs ending on a slash may no longer work as expected and may now lead to a 404. Examples of affected URLs are the Spring Boot actuator URLs. For further details see https://github.com/spring-projects/spring-framework/issues/28552.

Spring Boot 3 removed the support for registering auto-configurations in `spring.factories` using the `org.springframework.boot.autoconfigure.EnableAutoConfiguration` key. Instead, use the `META-INF/spring/org.springframework.boot.autoconfigure.AutoConfiguration.imports` file for registering auto-configurations. For further details see the official upgrade guide.

Spring Boot 3 removed the `org.springframework.beans.factory.annotation.Required` annotation. Use constructor injection or an implementation of `org.springframework.beans.factory.InitializingBean` instead.

7.12.3 Removed Usages of Apache HTTP Components

All usages of Apache HTTP Components 4.x (`org.apache.httpcomponents:httpclient` and `org.apache.httpcomponents:httpcore`) have been removed from core and Blueprint sources. The Apache HTTP client implementations has been replaced with Spring's RestTemplate.

Please note that the HTTP client implementation of the RestTemplate depends on the presence of HTTP client implementations in the application's class path. For further information see the API documentation of `ClientHttpRequestFactories#get(ClientHttpRequestFactorySettings)`.

Although the HTTP Components 4.x are not directly used by CMCC 12 code anymore, their dependency versions are still managed in the project, because they are still required by a number of frameworks that are used by the CoreMedia CMS implementations.

7.12.4 Replacement/Removal of Deprecated Code

Need changes when you expect certain logs from `hox.corem.server.Server`

The location of loggers for certain classes changed. The following classes are now using a logger in their own class instead of the logger of the `hox.corem.server.Server` class:

- `hox.corem.server.ResourceImpl`
- `hox.corem.server.DocumentTypeImpl`
- `hox.corem.server.FolderImpl`
- `hox.corem.server.RepositoryImpl`
- `hox.corem.server.ListenerManager`
- `hox.corem.server.CapUserManager`
- `hox.corem.server.ResourceCache`
- `hox.corem.server.SessionManager`
- `hox.corem.server.SortingServiceFactory`

Common:

- `com.coremedia.publisher.importer.XmlPropertyTransformer#getFilter` now also throws `NoSuchMethodException` and `InvocationTargetException`

Middle:

- `com.coremedia.cap.util.ContentCacheKey` now extends `com.coremedia.cache.util.ObjectCacheKey` instead of `com.coremedia.cap.util.ObjectCacheKey`

7.12.5 Component XML Files Converted to Sprint Boot Auto Configuration

Changes only necessary when you imported component XML files before

When you have imported the component XML files, you now have to import the configuration classes.

In the course of migrating from Spring XML to Java configurations, some XML files were removed in favor of their replacement Java Spring configuration classes. The details can be found below:

- removed bpbase-links-postprocessor.xml (import BlueprintLinksPostprocessorsConfiguration instead)
- removed bpbase-context-finder-services.xml (import NavigationContextFinderConfiguration instead)
- removed handler-services.xml (import CaeHandlerServicesConfiguration instead)
- removed link-services.xml (import CaeLinkServicesConfiguration instead)
- removed user-services.xml (import UserServicesConfiguration instead)
- removed cap-xliff-service.xml (import CapXliffConfiguration instead)
- removed mimetype-service.xml (import MimeTypeServiceConfiguration instead)
- removed com.coremedia.cap.test.xmlrepo.XmlRepoResources (import Java configuration classes instead)

In the course of updating from spring boot 2.7 to 3.2 a number of configuration XML files and configuration classes had to be adjusted:

- Bean `translate.xliff.translatableExpressions` provided by class `com.coremedia.translate.item.TranslatablePredicateConfiguration` was renamed to `translateXliffTranslatableExpressions`
- `com.coremedia.blueprint.studio.topicpages.rest.CustomTopicPagesConfiguration` was renamed to `CustomTopicPagesAutoConfiguration`
- `com.coremedia.blueprint.analytics.elastic.rest.ESALXStudioConfiguration` was renamed to `ESALXStudioAutoConfiguration`
- `com.coremedia.blueprint.taxonomies.TaxonomyConfiguration` was renamed to `TaxonomyAutoConfiguration`

- `com.coremedia.blueprint.studio.rest.taxonomies.TaxonomyStudioConfiguration` was renamed to `TaxonomyStudioAutoConfiguration`
- `com.coremedia.blueprint.studio.rest.intercept.InterceptorsStudioConfiguration` was renamed to `InterceptorsStudioAutoConfiguration`
- XML configuration file `com/coremedia/blueprint/common/multisite/translation-config.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/studio-in-memory-cap-list.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/es-alx-common.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/es-alx-contentbeans.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/es-alx-retrieval.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/caefeeder-blueprint.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/caefeeder-services.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/livecontext-contentfeeder.xml` was removed without replacement
- XML configuration file `META-INF/coremedia/elastic-worker.xml` was removed without replacement

The CoreMedia Component Loader is deprecated in favor of Spring Boot auto configurations. Many Spring component XML files have been converted to Spring Boot auto configuration classes. Several non-component Spring XML files have been converted to Spring configuration classes in that course. Also, existing configuration classes have been renamed and component properties files have been renamed to ordinary properties files. The detailed changes are listed below.

Migration from CoreMedia Component Loader to Spring Boot auto configurations:

- component-lc-asset.xml —> LcCaeAssetAutoConfiguration
- component-elastic-social.xml → ElasticSocialCaeAutoConfiguration
- component-alx-cae.xml → AlxCaeAutoConfiugration
- component-am-cae.xml → AMCaeAutoConfiguration
- component-blueprint-cae.xml → CaeBaseComponentAutoConfiguration
- component-corporate-cae.xml → CorporateCaeAutoConfiguration
- component-alx-google-cae.xml → AlxGoogleCaeAutoConfiguration
- component-personalization.xml → P13NCaeAutoConfiguration
- component-p13n-preview-cae.xml → P13NPreviewCaeAutoConfiguration
- component-alx-p13n-cae.xml → P13NAlxCaeAutoConfiguration
- component-cae.xml → CaeComponentAutoConfiguration
- component-cap-client.xml → CapClientComponentAutoConfiguration

- component-base.xml → CustomizerAutoConfiguration

Migration of Spring XML to Spring Java configuration:

- livecontext-handler-interceptors.xml → LcCaeInterceptorsConfiguration
- component-lc-elastic-social.xml → LcElasticSocialCaeAutoConfiguration
- component-lc-p13n-cae.xml → LcP13NCaeAutoConfiguration
- component-lc-preview-cae.xml → LcPreviewCaeAutoConfiguration
- component-lc-p13n-preview-cae.xml → LcP13NPreviewCaeAutoConfiguration
- livecontext-preview-links.xml → LcPreviewCaeLinkConfiguration
- livecontext-preview-handler-interceptors.xml → LcPreviewCaeInterceptorsConfiguration
- personalization-context.xml → P13NCaeContextConfiguration
- p13n-preview-cae-context.xml → P13NPreviewCaeContextConfiguration
- personalization-collection.xml → ContextCollectionConfiguration

Removal of Spring XML configuration files:

- removed lc-asset-handlers.xml (import LcCaeAssetHandlersConfiguration instead)
- removed blueprint-i18n.xml (import com.coremedia.blueprint.cae.config.BlueprintI18nCaeBaseLibConfiguration instead)
- removed blueprint-l10n.xml (import com.coremedia.blueprint.cae.config.BlueprintL10nCaeBaseLibConfiguration instead)
- removed blueprint-search.xml (import com.coremedia.blueprint.cae.config.BlueprintSearchCaeBaseLibConfiguration instead)
- removed blueprint-richtextfilters.xml (import com.coremedia.blueprint.cae.config.BlueprintRichtextFiltersConfiguration instead)
- removed livecontext-validation.xml (import LcCaeValidationConfiguration instead)
- removed livecontext-links.xml (import CommerceLinkConfiguration instead)
- removed livecontext-fragment.xml (import LcCaeFragmentConfiguration instead)
- ec-cae-lib.xml was resolved. Its imports were transferred to CorporateCaeAutoConfiguration and LcCaeAutoConfiguration.
- removed livecontext-preview-hybrid.xml. It only contained a customizer, which was migrated to LcPreviewCaeAutoConfiguration.
- removed blueprint-links.xml (import com.coremedia.blueprint.cae.config.BlueprintLinksCaeBaseLibConfiguration instead)
- removed personalization-interceptors.xml (import com.coremedia.blueprint.personalization.config.P13NInterceptorsConfiguratio instead)
-
  ```
  removed p13n-preview-cae-context-.xml (import
          com.coremedia.blueprint.personalization.preview.config.P13NPreviewCaeContextConfiguration instead)
  ```
- removed bpbase-sitemodel.xml (import com.coremedia.blueprint.base.multisite.BlueprintMultisiteModelConfiguration instead)
- removed view-development-services.xml (import com.coremedia.objectserver.view.config.CaeViewDevelopmentServicesConfiguration instead)

- removed view-error-services.xml (import com.coremedia.objectserver.view.config.CaeViewErrorServicesConfiguration instead)
- removed customizer-services.xml (import com.coremedia.springframework.customizer.CustomizerConfiguration instead)

Renamed Spring Java configurations:

- LcCaeAssetConfiguration → LcCaeAssetHandlersConfiguration
- CaeBaseComponentConfiguration → CaeBaseComponentAutoConfiguration
- LiveContextPreviewCaeConfiguration → LcPreviewCaeAutoConfiguration
- CapClientComponentConfiguration → CapClientComponentAutoConfiguration

Miscellaneous:

-
  ```
  removed component-blueprint-cae.properties as it only contained
        properties set to the default values or commented out properties.
  ```
- removed abstract bean 'storeContextInterceptor' as there is no such thing as abstract beans in Java config.
- removed component-livecontext.properties → livecontext.properties (imported in LcCaeAutoConfiguration)
- migrated bean definitions from CaeComponentConfiguration to CaeComponentAutoConfiguration and removed it
- removed com.coremedia.personalization.context.collector.LicenseHelper from public API
- removed com.coremedia.cms:cae-util module. Consequently, uapi-services.xml and CaeUtilConfiguration were removed as well. The beans were migrated to ContentBeanServicesConfiguration.

7.12.6 Removal of Pre-created Directories in coremedia/java-application-base Image

Starting with `coremedia/java-applicaition-base:2.1-cm-17.*`, the base image no longer precreates the following directories:

- `/coremedia/tools/var/logs`
- `/coremedia/corem-home/var/tmp`
- `/coremedia/log`

7.13 Deployment Changes

Which changes have been made to the deployment of the CoreMedia system?

7.13.1 APPLICATION_OPTS Removed from Base Image

Starting with `coremedia/java-application-base:2.1-cm-17.*` the environment variable toggle `APPLICATION_OPTS` has been removed from the base image, please use `_JAVA_OPTIONS` instead as intended by the JVM.

7.13.2 Volume Definitions Moved to Application Module or Deployment

Starting with `coremedia/java-application-base:2.1-cm-17.*` the base image only defines the new default for `java.io.tmpdir`, `/tmp` as a volume. Volumes should be defined in the deployment code matching the configured paths for ephemeral and persistent state paths.

This change is one of many to consolidate the image creation to make the containers immutable with a readonly rootfs.

7.13.3 The coremedia/java-application-base Image with Corretto JRE

The `coremedia/java-application-base` image now also provides a JRE based Corretto image. The image tag suffix is `corretto-jre`. If there are Java modules missing for your use case, there is still the JDK based image with the suffix `corretto-jdk` or the Eclipse Temurin based images with the prefixes `temurin-jre` or `temurin-jdk`.

7.13.4 Change of the Default java.io.tmpdir path

The `coremedia/java-application-base` images starting with `2.1-cm-17.*` will no longer set `java.io.tmpdir` to `/var/tmp` by default but instead to `/tmp` to match the default used in most JVM frameworks like Apache Tomcat uses it for its root dir when embedded in Spring Boot applications. This is a consolidation to make CoreMedia container based applications easier to configure when securing it to have a readonly rootfs.

7.13.5 Prometheus jmx-exporter Agent Disabled by Default

Use the Spring Boot Prometheus actuator endpoint to scrape metrics instead. You can still enable the exporter using `PROMETHEUS=true` but you also need to expose the `8199` port for the Prometheus service discovery.

To enable the Spring Boot Prometheus actuator endpoint, make sure that it is included in the list of exposed endpoints, configured by `management.endpoints.web.exposure.include`.

7.13.6 SPRING_BOOT_EXPLODED_APP Removed from Base Image

NOTE

Only for information. No action required.

Starting with `coremedia/java-application-base:2.1-cm-17.*` the environment variable toggle `SPRING_BOOT_EXPLODED_APP` has been removed from the base image. Images build using Google Jib Maven plugin are always exploded, making this toggle obsolete.

[CMS-23618]

7.13.7 The environment variable SPRING_PROFILES has been removed from the base image and the compose setup

Starting with `coremedia/java-application-base:2.1-cm-17.*`, using the environment variable `SPRING_PROFILES` is deprecated. It will still work but you should migrate and use the `SPRING_PROFILES_ACTIVE` environment variable.

[CMS-23613]

7.13.8 coremedia/java-application-base is now Based on amazonlinux:2023

Starting with `coremedia/java-application-base:2.1-cm-17*` the image is based on `public.ecr.aws/amazonlinux/amazonlinux:2023-minimal`.

[CMS-23330]

7.13.9 Port 8199 Removed from the Exposed Ports by Default

Only in case the container exposes the jmx-exporter metrics endpoint by setting `PROMETHEUS=true`, the container should expose the port. Because the jmx-exporter is now disabled by default, this port has been removed from the expose list, so it does not get accidentially scraped by prometheus.

If you want to continue to use the jmx-exporter, you should expose the port in your deployment configuration.

[CMS-22787]

7.14 Frontend Workspace Changes

This section describes changes in the CoreMedia Frontend workspace.

7.14.1 Thirdparty Upgrade: Upgraded Node-Sass to 9.0.0

Node-Sass has been updated to work with *NodeJS* 20.x.

7.14.2 Thirdparty Upgrade: Upgraded Webpack to 5.90.3

Upgrade Webpack necessary

!

Webpack was upgraded to 5.90.3. This also includes upgrades to all associates loader and plugin packages in `theme-utils` as well as `node-sass`. This change was necessary to be able to upgrade to *NodeJS* 20 LTS.

Upgrade steps:

Please run `pnpm remove -r cross-env` and adjust the following entry of your themes' `package.json` files from:

```
From:
  "scripts": {
    ...
    "build": "cross-env NODE_OPTIONS=--openssl-legacy-provider webpack",
    ...
  }
```

back to

```
To:
  "scripts": {
    ...
    "build": "webpack",
    ...
  }
```

Please also make sure that the `webpack` entry is adjusted to `^5.90.3`.

If you have made any customizations to the `webpack.config.js` of your themes, you might need to make further adjustments like adjusting configuration or upgrading your used webpack loaders and plugins for compatibility. The offical upgrade guide can be found here: To v5 from v4 | webpack.

7.14.3 Removed deprecated CSS/JS property "ieExpression"

Changes only necessary when ieExpression was used

The doctype property `ieExpression` of `CMAbstractCode` has been deprecated in release 2110.1. All usages in Studio, CAE and Frontend are removed now. To avoid migration issues, the doctype property still exists in the CoreMedia content type model.

7.14.4 Third-Party Update: Upgraded jQuery to version 3.7.1

Changes only necessary when using jQuery in themes

Updated jQuery to 3.7.1 in the blueprint and in the frontend workspace. If you are using jQuery in your themes, please update the dependency in the package.json of the theme too to avoid importing both jquery versions.

Upgrade steps:

Run `pnpm -r up jquery@3.7.1` in your frontend workspace.

[CMS-23970]

7.14.5 Modernized eslint and prettier in Frontend Workspace

Changes only necessary when using eslint and prettier

We updated and integrated eslint and prettier (via eslint-plugin) to the whole frontend workspace. It is enabled and configured for all themes and bricks via a shared configuration. It is integrated in the frontend tooling too. You can check and lint all files via `pnpm lint`. All blueprint source files have been linted.

Upgrade steps:

If you don't want to use eslint and prettier, everything should work as before. Otherwise run the following command in your custom themes and bricks: `pnpm add -D eslint @coremedia/eslint-config-frontend` Also add the following entry to the scripts block in the `package.json`: `"lint": "eslint --fix \"src/*/.js\""`

(CMS-23949)

7.15 Command Line Tool Changes

This section describes changes applied to the command line tools.

7.15.1 Deleted Deprecated Options of "Query" Commandline Tools

Changes only Necessary when Query Tool was Used with Parameters

The **query** tool can be used to start a synchronous, structured query against the content repository.

The deprecated *`paths`* and *`versions`* (short option: e) options of the **query** command line tool were removed. Use the *`path`* and *`versionquery`* parameter instead.