



Multi-Site Manual

CoreMedia Content Cloud – v13

Copyright CoreMedia GmbH © 2026

CoreMedia GmbH

Altes Klöpperhaus, 5. OG

Rödingsmarkt 9

20459 Hamburg

International

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

Germany

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

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

Licenses and Trademarks

All trademarks acknowledged.
July 22, 2026 (Release 2512.1)

1. Preface	1
1.1. Audience	2
1.2. See Also	3
1.3. CoreMedia Services	4
1.3.1. Registration	4
1.3.2. CoreMedia Releases	5
1.3.3. Documentation	6
1.3.4. CoreMedia Training	9
1.3.5. CoreMedia Support	9
1.4. Typographic Conventions	11
2. Overview	13
3. Designing Your Multi-Site Experience	15
3.1. Designing Site Hierarchies	16
3.1.1. Translation and Synchronization	16
3.1.2. Variants of Site Hierarchies	17
3.1.3. Guiding Questions	19
3.1.4. Recommended Sites Hierarchy	20
3.2. Designing Locales	22
3.2.1. Defining Valid Language Tags	22
3.2.2. Adding Locales to CoreMedia Studio	26
3.2.3. Example: The Global Site Locale	26
4. Multi-Site Challenges	28
4.1. General Multi-Site Content Integrity Note	29
4.2. Concurrent Editing Feature	30
4.3. Deriving a Site	31
4.4. CleanInTranslation Challenges	33
4.5. Referrer Relations in Localization	43
4.6. General Multi-Site Fixes	45
4.7. Overview of Possible Pitfalls	46
4.8. Preconditions for Comprehensive Changes	53
4.9. Changing the Site Type	55
4.9.1. The Site Type	55
4.9.2. Apply Type Change	56
4.10. Changing the Site Hierarchy	57
4.10.1. Example Hierarchy	58
4.10.2. Moving Sites – General Concept	58
4.10.3. Apply New Hierarchy	59
Glossary	65
Index	73

List of Figures

- 2.1. Master content and Derived content 13
- 3.1. Sites: Language First 17
- 3.2. Sites: Country First 18
- 4.1. Conflict Site vs. Content Locale 51

List of Tables

- 1.1. CoreMedia manuals 6
- 1.2. Typographic conventions 11
- 1.3. Pictographs 12
- 3.1. Subtags Represented in Display Name 23
- 4.1. List of Multi-Site Challenges 46
- 4.2. Site Hierarchy Transformation 58

List of Examples

- 3.1. Creating Locale with Locale.Builder 25
- 4.1. Standard Activity in Workflow Server Log 33
- 4.2. Warning in Workflow Server Log 34
- 4.3. Dump Localization Process 34
- 4.4. Dump Output 34
- 4.5. Standard Activity in Workflow Server Log on Debug Level 36
- 4.6. Content Cleanup Preparation 36
- 4.7. Content Cleanup Execution 36
- 4.8. General Issues Report 37
- 4.9. Issue Output for DERIVED_DELETED 39
- 4.10. Issue Output for DERIVED_DESTROYED 40
- 4.11. Issue Output for DERIVED_MASTER_MISMATCH 40
- 4.12. Issue Output for DERIVED_NO_MASTER 41
- 4.13. Issue Output for MASTER_DESTROYED 41
- 4.14. Translation Settings Struct 55

1. Preface

The Multi-Site Manual provides an overview of the CoreMedia Multi-Site feature. It describes different options to design your site hierarchy and gives some guidance to avoid common pitfalls.

- [Chapter 2, Overview \[13\]](#) describes the main concepts of multi-site.
- [Chapter 3, Designing Your Multi-Site Experience \[15\]](#) gives you guidelines to design your site hierarchy and custom locales.
- [Chapter 4, Multi-Site Challenges \[28\]](#) describes common problems and solutions when using multi-site in an inappropriate way.

Multi-Site is not Multi-Site

If you know CoreMedia CMS from the early days, you may know a different feature when talking about Multi-Site. It was about a feature where you were able using multiple *Master Live Servers* as publication targets. Nowadays, this feature is called *CoreMedia Multi-Master Management*.

To get to know more about this feature, have a look at [Section 2.3, “Multi-Master Publishing”](#) in *Content Server Manual*.



1.1 Audience

This manual is intended for architects, developers and editors who want to work with CoreMedia Multi-Site or who want to learn about the concepts of this feature. The reader should be familiar with content management in *CoreMedia Content Cloud*.

1.2 See Also

This manual describes guidelines for the design of your site hierarchy and describes pitfalls in working with the multi-site feature. However, other aspects of multi-site are featured in various other CoreMedia manuals.

For further reference have a look at the following additional resources:

- [Studio User Manual](#)
 - [Section 2.10, “Multi-Site and Multi-Language”](#) in *Studio User Manual*
 - [Section 3.3.12, “Configuring Workflow Validation”](#) in *Studio User Manual*
 - [Section 4.7.4.3, “Comparing Translation”](#) in *Studio User Manual*
 - [Section 4.7.4, “Translating Content”](#) in *Studio User Manual*
 - [Section 4.7.5, “Synchronizing Content”](#) in *Studio User Manual*
 - [Section 4.7.6.4, “Localizing a Project”](#) in *Studio User Manual*
- [Studio Developer Manual](#)
 - [Section 9.28.9, “Translation Workflow Specifics”](#) in *Studio Developer Manual*
 - [Section 9.28.10, “Synchronization Workflow Specifics”](#) in *Studio Developer Manual*
- [Blueprint Developer Manual](#)
 - [Section 5.5, “Localized Content Management”](#) in *Blueprint Developer Manual*
 - [Section 5.5.3.5, “Translation Workflow”](#) in *Blueprint Developer Manual*
 - [Section 5.7.3, “Deriving Sites”](#) in *Blueprint Developer Manual*
 - [Section 5.7.4, “Synchronization Workflow”](#) in *Blueprint Developer Manual*
- [Content Server Manual](#)
 - [Section 3.13.1.11, “Validate Multi-Site”](#) in *Content Server Manual*

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" \[4\]](#) 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" \[4\]](#) describes how to register for the usage of the services.
- [Section 1.3.2, "CoreMedia Releases" \[5\]](#) describes where to find the download of the software.
- [Section 1.3.3, "Documentation" \[6\]](#) describes the CoreMedia documentation. This includes an overview of the manuals and the URL where to find the documentation.
- [Section 1.3.4, "CoreMedia Training" \[9\]](#) describes CoreMedia training. This includes the training calendar, the curriculum and certification information.
- [Section 1.3.5, "CoreMedia Support" \[9\]](#) describes the CoreMedia support.

1.3.1 Registration

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

1.3.2 CoreMedia Releases

Downloading and Upgrading the Blueprint Workspace

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

<https://releases.coremedia.com/cmcc-13>

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

NOTE

If you encounter a 404 error then you are probably not logged in at GitHub or do not have sufficient permissions yet. See [Section 1.3.1, "Registration" \[4\]](#) 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 3.1, "Prerequisites"](#) in *Blueprint Developer Manual*.

npm packages

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

<https://repository.coremedia.com/nexus/repository/coremedia-npm/>

The `.npmrc` file must be configured to be able to use 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" \[9\]](#)) 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
Blueprint Developer Manual	Developers, architects, administrators	This manual gives an overview over the structure and features of <i>CoreMedia Content Cloud</i>. It describes the content type model, the <i>Studio</i> extensions, folder and user rights concept and many more details. It also describes administrative tasks for the features. It also describes the concepts and usage of the project workspace in which you develop your CoreMedia extensions. You will find a description of the Maven structure, the virtualization concept, learn how to perform a release and many more.
Connector Manuals	Developers, administrators	This manual gives an overview over the use cases of the eCommerce integration. It describes the deployment of the Commerce Connector and how to connect it with the CoreMedia and eCommerce system.
Content Application Developer Manual	Developers, architects	This manual describes concepts and development of the <i>Content Application Engine (CAE)</i>. You will learn how to write Freemarker templates that access the other CoreMedia modules and use the sophisticated caching mechanisms of the CAE.
Content Server Manual	Developers, architects, administrators	This manual describes the concepts and administration of the main CoreMedia component, the <i>Content Server</i>. You will learn about the content type model which lies at the heart of a CoreMedia system, about user and rights management, database configuration, and more.

Manual	Audience	Content
Deployment Manual	Developers, architects, administrators	This manual describes the concepts and usage of the CoreMedia deployment artifacts. That is the deployment archive and the Docker setup. You will also find an overview of the properties required to configure the deployed system.
Elastic Social Manual	Developers, architects, administrators	This manual describes the concepts and administration of the <i>Elastic Social</i> module and how you can integrate it into your websites.
Frontend Developer Manual	Frontend Developers	This manual describes the concepts and usage of the Frontend Workspace. You will learn about the structure of this workspace, the CoreMedia themes and bricks concept, the CoreMedia Freemarker facade API, how to develop your own themes and how to upload your themes to the CoreMedia system.
Headless Server Developer Manual	Frontend Developers, administrators	This manual describes the concepts and usage of the <i>Headless Server</i> . You will learn how to deploy the Headless Server and how to use its endpoints for your sites.
Multi-Site Manual	Developers, Multi-Site Administrators, Editors	This manual describes different options to design your site hierarchy with several languages. It also gives guidance to avoid common pitfalls during your work with the multi-site feature.
Operations Basics Manual	Developers, administrators	This manual describes some overall concepts such as the communication between the components, how to set up secure connections, how to start application.
Search Manual	Developers, architects, administrators	This manual describes the configuration and customization of the <i>CoreMedia Search Engine</i> and the two feeder applications: the <i>Content Feeder</i> and the <i>CAE Feeder</i> .
Studio Developer Manual	Developers, architects	This manual describes the concepts and extension of <i>CoreMedia Studio</i> . You will learn about the underlying concepts, how to use the devel-

Manual	Audience	Content
		opment environment and how to customize <i>Studio</i> to your needs.
Studio User Manual	Editors	This manual describes the usage of <i>CoreMedia Studio</i> for editorial and administrative work. It also describes the usage of the <i>Native Personalization</i> and <i>Elastic Social</i> GUI that are integrated into <i>Studio</i> .
Studio Benutzerhandbuch	Editors	The Studio User Manual but in German.
Supported Environments	Developers, architects, administrators	This document lists the third-party environments with which you can use the CoreMedia system, Java versions or operation systems for example.
Unified API Developer Manual	Developers, architects	This manual describes the concepts and usage of the <i>CoreMedia Unified API</i> , which is the recommended API for most applications. This includes access to the content repository, the workflow repository and the user repository.
Utilized Open Source Software & 3rd Party Licenses	Developers, architects, administrators	This manual lists the third-party software used by CoreMedia and lists, when required, the licence texts.
Workflow Manual	Developers, architects, administrators	This manual describes the <i>Workflow Server</i> . This includes the administration of the server, the development of workflows using the XML language and the development of extensions.

Table 1.1. CoreMedia manuals

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

Email: documentation@coremedia.com

1.3.4 CoreMedia Training

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

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

<https://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:

<https://support.coremedia.com/>

Do not forget to request further access via email after your initial registration as described in [Section 1.3.1, "Registration" \[4\]](#). The support email address is:

Email: support@coremedia.com

Create a support request

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

Support request

- Which CoreMedia component(s) did the problem occur with (include the release number)?
- Which database is in use (version, drivers)?
- Which operating system(s) is/are in use?
- Which Java environment is in use?

- Which customizations have been implemented?
- A full description of the problem (as detailed as possible)
- Can the error be reproduced? If yes, give a description please.
- How are the security settings (firewall)?

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

To put it in a nutshell, CoreMedia needs:

Support checklist

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

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

Log files

Which Log File?

In most cases at least two CoreMedia components are involved in errors: the *Content Server* log files together with the log file from the client. If you know exactly what the problem is, solving the problem becomes much easier.

Where do I Find the Log Files?

By default, 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 *kubectrl* command-line client in a Kubernetes environment you can use the **kubectrl logs** command to access the logs. For a detailed instruction of how to use the command, see [kubectrl logs](#). Make sure to enable the timestamps using the `--timestamps` flag.

```
kubectrl logs --timestamps <pod>
```

1.4 Typographic Conventions

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

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

Table 1.2. Typographic conventions

In addition, these symbols can mark single paragraphs:

Pictograph	Description
	Tip: This denotes a best practice or a recommendation.
	Warning: Please pay special attention to the text.
	Danger: The violation of these rules causes severe damage.

Table 1.3. Pictographs

2. Overview

When talking about CoreMedia Multi-Site, it is a valid assumption that it is about translation, that is, delivering content in different languages. But CoreMedia Multi-Site is more than that.

CoreMedia Multi-Site not only supports translation of content, but also localization, which is, that you may deliver sites having different languages and different structures. For example, you may want to add a banner for Thanksgiving holiday in your American site which is missing in all other sites.

CoreMedia believes, that with the provided model, you will be able to easily manage content in many languages, while having the flexibility to adapt your site to local needs. You may decide to translate content on your own, or you will be able to let translation agencies do the translation for you.

CoreMedia Multi-Site will enable you addressing your customers with a customized web presence adapted to each country, while on the other hand providing an easy way to manage and adapt these customizations.

The following paragraphs give you a first introduction into the main concepts of the multi-site feature.

Master and Derived

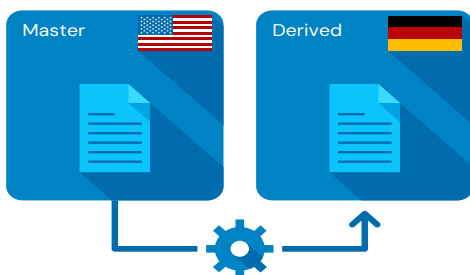


Figure 2.1. Master content and Derived content

For each language you want to present on your website, there is a dedicated set of content items, each annotated with the language they belong to. Content items which represent the same content but in different languages, are linked to each other. These links are annotated by version identifiers, so that you will

always know, if you need to translate a content again, because its sibling got updated. Later in this manual we will refer to these relations as **master** and **derived** as for a better overview, these siblings are organized hierarchically.

Sites

Each of those content sets is kept in a dedicated folder, so that you will have a folder for all English contents, a folder for all Arabic contents, and another one for all German contents. Each of these folders represent a **site**.

Workflows

To transfer a content set from English to Arabic, you will use CoreMedia workflows. These automatic processes will guide you through the translation process and assist, for example, not to forget to translate a linked content, which is referred from the current content you are going to translate.

These workflows will also take care of adapting properties of your contents, which are not to be translated, but should be changed, as soon as the original content changed. Most prominent example are links. When you add a link to your content and you are going to translate the content, the link will be added to your target content automatically.

3. Designing Your Multi-Site Experience

In this chapter you will get to know the required details to successfully design your initial Multi-Site experience and you will also learn which design decisions will be difficult to adapt afterwards.

A core value of *CoreMedia Content Cloud* is its flexibility. It easily adapts to changed requirements. While CoreMedia Multi-Site provides similar flexibility in various aspects, there are some aspects, which require upfront planning and are not easy to adjust later on. This especially applies to the hierarchy of sites.

3.1 Designing Site Hierarchies

This section guides you through the upfront design of your sites hierarchy with focus on the recommended hierarchy as used by many of CoreMedia's customers.

The decision on the hierarchy of your sites within CoreMedia Multi-Site will influence your editorial processes. As changing the site hierarchy after initial kickoff is not an easy task, it is recommended to spend some thoughts upfront on the hierarchy in the design phase of your project.

NOTE

The structure of the sites is strictly hierarchical. All derived sites can only have one site as their master. The propagation of content items is always from top to bottom, that is, from master to derived.



3.1.1 Translation and Synchronization

The variants of possible site hierarchies are based on two different types of master-derived site relations:

- Translation
- Synchronization

Translation is the basic type of CoreMedia Multi-Site: Having a master language, such as *English (United States)* you create a derived site such as *German (Germany)* which represents your site in Germany. To transfer a content from your master site to your derived site requires translation.

 is the icon used to derive a translated site. In the figures of [Section 3.1.2, "Variants of Site Hierarchies" \[17\]](#) it denotes translated sites.

Synchronization is an alternative type of CoreMedia Multi-Site: Having a master language, such as *English (United States)* you create a derived site such as *English (New Zealand)* which represents your site in New Zealand. To transfer a content from your master site to your derived site involves an automated process similar to copy and paste.

 is the icon used to derive a synchronized site. In the figures of [Section 3.1.2, "Variants of Site Hierarchies" \[17\]](#) it denotes synchronized sites.

3.1.2 Variants of Site Hierarchies

The following are variants of site hierarchies, which are most commonly used, sometimes even in mixed forms:

Language First (Recommended)

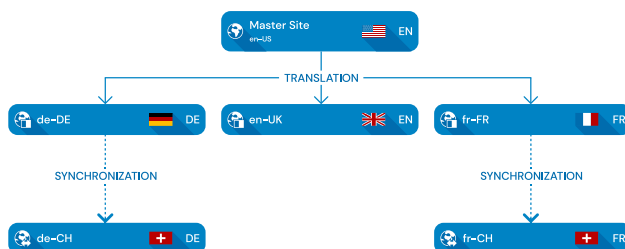


Figure 3.1. Sites: Language First

This approach focuses on minimizing costs for translation by only having to translate content into each language once. It enables rolling out campaigns quickly and with little manual work once the content has been translated.

On the other hand, it is potentially more difficult to apply structural differences that are only relevant for sites that belong to one country. These sites are distributed across the hierarchy and consequently changes need to be applied manually to each one. Translation and synchronization works best when the site structure is mostly the same. Introducing [section "Abstract Sites" \[18\]](#) as master sites can simplify this manual process.

From a given master site you can perform translations to several derived sites. If a given language is applicable for multiple countries and for example the navigation structure needs to be adapted in these countries, the translations are synchronized to country-centric sites.

In [Figure 3.1, "Sites: Language First" \[17\]](#) the sites for Switzerland are once located beneath the site having locale *German (Germany)* and once beneath *French (France)*, as it is the language which decides on the structure prior to the country.

Country First

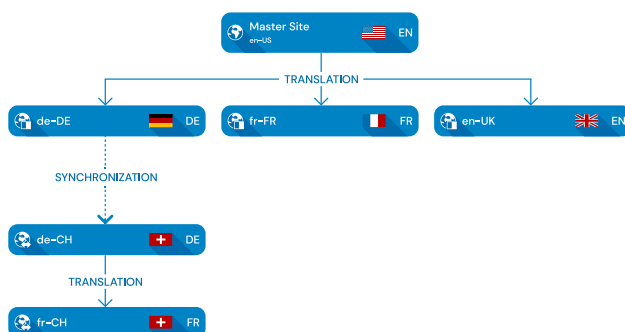


Figure 3.2. Sites: Country First

In contrast to the Language First approach, this approach optimizes for easily propagating country-specific changes. This might require translating content multiple times though.

From a given master, you synchronize (or translate) to a site per country you want to deliver your website to. If you want to deliver your website in various languages in a given country, you must create a translation site for each of these languages.

In Figure 3.2, “Sites: Country First” [18] the sites for Switzerland are both located beneath the site having locale *German (Germany)*, first synchronized to *German (Switzerland)* as it has the same language as the German site, then translated to *French (Switzerland)*.

Alternatively, you may have placed the sites for Switzerland below *French (France)*, first synchronizing to *French (Switzerland)* and then translating to *German (Switzerland)*.

Abstract Sites

Abstract sites are an approach to consolidate content globally. An abstract site provides a pool of all editorial content, and a common structure for all its derived sites. There are real-world examples where multiple abstract sites helped to minimize the editorial workload even further by integrating the market-specific changes.

Take for example an abstract master *English (North America)* that is derived from another abstract master site *English (World)*. Both will never be seen by the end-users but editors can integrate changes that apply to North America

in the corresponding abstract site. As long as the (abstract) master sites and the derived sites use the same language, editors can use the synchronization workflow with little manual work to propagate changes and new content.

Again, an abstract site is a master site, which is not published at all, thus, it is not available as a website. This abstract site holds *all* content items, which are relevant to at least one of the derived sites. You synchronize and translate its content to these sites which are derived from the abstract site.

Even if a content is only necessary for one specific site, create it in the abstract master. You can later easily reuse this content in other sites, if requirements change.

As every site needs a valid locale in terms of a [IETF BCP 47](#) language tag, you may require creating some artificial language tag, as for example `en-US-utty-abstract`, which will be displayed to *CoreMedia Studio* users as *English (United States, ty/: abstract)*. For details see [Section 3.2, "Designing Locales" \[22\]](#).

3.1.3 Guiding Questions

The following guiding questions can help you find a hierarchy which matches best:

- **How similar are the sites regarding their structure, especially their navigation structure?**

For the recommended *Language First Hierarchy* the navigation structure should be similar throughout the hierarchy.

- **How many languages are spoken in the various countries I want to deliver my website to?**

The more often a given language is chosen throughout the site hierarchy, the more the *Language First Hierarchy* will fit your needs.

- **Are there different external data references (such as product IDs) which differ per country?**

If these references are strongly related to the country, you may be better off with the *Country First Hierarchy*. Otherwise, given [the example above of the Swiss sites](#), you would have to manually synchronize external references in the *Language First Hierarchy* between *German (Switzerland)* and *French (Switzerland)*.

- **What is the ratio of textual changes versus structure (for example, navigation structure) changes you are expecting?**

The more you update textual content, the more translation tasks will occur, and the more the *Language First Hierarchy* will fit your needs.

- **What are the costs of structure (navigation) changes versus translations?**

The lower the cost for translation are, the less important it is which hierarchy you are choosing. The *Language First Hierarchy* pays off as soon as translation costs reach a considerable amount.

- **What is the structure of my editorial team?**

If your editorial team is spread across various countries, each responsible for a given country, the *Country First Hierarchy* may be the better fit, as you will reduce dependencies between editorial teams.

If your editorial team is tailored by languages, the *Language First Hierarchy* is obviously the better fit.

3.1.4 Recommended Sites Hierarchy

CoreMedia recommends the *Language First Hierarchy* for several reasons:

- **Given Optimizations**

CoreMedia Content Cloud is designed supporting especially the *Language First Hierarchy*. Also, in the future CoreMedia will further improve the editorial experience to overcome the shortcomings compared to the *Country First* approach.

- **Less Translation Costs**

In [Figure 3.1, “Sites: Language First” \[17\]](#) and [Figure 3.2, “Sites: Country First” \[18\]](#), a translation to French was only required once for the *Language First Hierarchy*, but twice for the *Country First Hierarchy*.

This is based on the fact of the strong hierarchical organization of sites within CoreMedia Multi-Site. Thus, for the *Country First Hierarchy*, you cannot transfer content from *French (France)* to *French (Switzerland)* guided by automated processes. Any manual intervention can easily break the internal state.

- **Easily Extensible**

If you want to reduce your time-to-market for your first site in China, you may want to set up a synchronized site having locale *English (China)*. Later on, you can derive additional translated sites for example for locales *Chinese (Simplified, China)* and *Cantonese (Simplified, China)*.

Changing Site Hierarchies is Expensive and Error-Prone

As stated early in this section, changing site hierarchies within a production system is expensive and error-prone. It requires a lot of manual intervention and most likely stopping any editorial actions until the migration is done.

Here is a glimpse of what needs to be done, by far not a complete list:

- **Finish Any Site-Specific Workflows:** When transforming your hierarchy, all translation and synchronization workflows must be finished.
- **Stop Editorial Work:** Editorial actions must be stopped. This includes manual as well as automatic editorial processes. All content must be checked in.
- **Ensure All Content is Up-To-Date:** There must be no more content that requires an update with regard to synchronization or translation.

After these preconditions are fulfilled, you will have to adjust all master-references (link and version number) starting from root to leaf sites, possibly adjust the type (translation or synchronization) of your site. And when you are done, you need to republish your sites.



3.2 Designing Locales

This section guides you through details in designing custom locales apart from the available locales as provided by your JDK.

Each site in CoreMedia Multi-Site requires a locale to be chosen. And within each hierarchy of sites, the chosen locale must be unique for one site within this hierarchy.

For special use cases such as [Abstract Sites](#) [18] it is possible to create custom locales. The following sections will guide you through the process of designing such locales.

Designing your custom locale comprises the following steps:

1. Define a valid and unique language tag as described in [Section 3.2.1, “Defining Valid Language Tags”](#) [22].
2. Add the new language tag to *Studio* as described in [Section 3.2.2, “Adding Locales to CoreMedia Studio”](#) [26].

A typical example for a custom locale is sketched in [Section 3.2.3, “Example: The Global Site Locale”](#) [26].

The World Region 001

IANA (Internet Assigned Numbers Authority) provides a region subtag 001 to represent the region *World*. This region subtag is registered at the [IANA Language Subtag Registry](#).

For some use-cases, like the aforementioned *Abstract Sites*, this region subtag may be a good alternative to custom locales.



3.2.1 Defining Valid Language Tags

Locales are denoted in content by IETF BCP 47 (Internet Engineering Task Force Best Current Practice no. 47) language tags as specified by [RFC 5646](#). A typical language tag consists of a language and country element such as `en-US` (*English (United States)*) or `de-DE` (*German (Germany)*). Valid language subtags are registered by IANA (Internet Assigned Numbers Authority) in the [IANA Language Subtag Registry](#). The most simple IETF BCP 47 language tags consist of the primary language subtag only, for example, `en` (*English*).

ietf bcp 47

Along with the criteria mentioned in the specification an additional restriction applies for locales used in context of *CoreMedia Studio*:

Locales must differ in their display name as generated by your JDK.

Thus, it is especially discouraged using so-called *Private Use Subtags* separated by reserved single-character subtag 'x' as they are not represented in the display name.

Display Name Localization Is Not Customizable

CoreMedia Content Cloud does not provide means for customizing localization of Locale display names. The display name is localized solely by your JDK, which in turn eases introducing new locales to your system without providing custom translations.



Display Names of Locales

If you are going to define a custom locale, it is important to ensure, that the display name of each locale offered to your editors differ regarding their display name. This section recommends possible options for designing a locale, so that customizations are visible in the display name.

In addition to primary language and region subtags the subtags listed in [Table 3.1, "Subtags Represented in Display Name" \[23\]](#) are known to be part of the display name and thus may be taken into account for creating customized locales for special needs.

Type	Description	Example
Script	Script subtags are defined in RFC 5646, Section 2.2.3. The length of a script subtag must be exactly four having only alphabetic characters. Script subtags are typically registered at IANA. One exception exists for script subtags <i>Qaaa</i> through <i>Qabx</i> which are for private use. While Java only checks for well-formed script subtags, it is recommended sticking to the specification.	en-Qaaa-US; displayed as <i>Eng-lish (Qaaa, United States)</i>

Type	Description	Example
Variant	Variant subtags are defined in RFC 5646, Section 2.2.5. A language tag may contain multiple variants separated by dashes. Each variant may only contain alphanumeric characters. In addition to the specification, current Java implementations limit the number of characters to eight at maximum. Just as scripts, variants are typically registered at IANA. There is no concept of private use variant subtags. Nevertheless, Java does not validate against registered variants.	<code>en-US-Variant</code> ; displayed as <i>English (United States, Variant)</i>
Extension: Unicode Locale Keyword	Extension subtags are defined in RFC 5646, Section 2.2.6. A special type of these extension subtags are <i>Private Use Subtags</i> defined in RFC 5646, Section 2.2.7. The separator for extensions must be registered at IANA. The separator 'x' is used for private use subtags. While the private use subtags provide most freedom choosing custom subtags, they are not displayed in the display name of the locale. Regarding Java CoreMedia recommends using so-called <i>Unicode Locale Keywords</i>, which allow a key-value based approach. These key-value pairs are prefixed with 'u' within the language tag. The following restrictions apply to the keyword: it must have a length of two characters and consist of alphanumeric characters. The following restrictions apply to the value: it may contain dash-separated values, where each single value has to be alphanumeric and have a length of three to eight (including) characters.	<code>en-US-u-ky-value</code> ; displayed as <i>English (United States, ky: value)</i>

Table 3.1. Subtags Represented in Display Name

Disclaimer: The actual behavior relies on your type and version of JDK.

Tip: Design Help Via Java's Locale

Defining a proper language tag may be a tedious task, like reading and understanding the IETF BCP 47 specification and aligning it with your JDK's behavior. To ease this task, Java provides a class `Locale.Builder` which may be used to create and validate language tags.

Using the `Locale.Builder` for your custom locales will guide you through the process of creating a valid language tag, understood by your JDK. In [Example 3.1, "Creating Locale with Locale.Builder" \[25\]](#) you find an example of how to create a locale with `Locale.Builder`. You may execute the code in an independent project, or as a temporary unit test, or within some online Java playground, to eventually get your desired valid language tag.

```
new Locale.Builder()
    // Base on existing locale.
    .setLocale(Locale.US)
    // ! Not part of Display Name !
    .setExtension(Locale.PRIVATE_USE_EXTENSION, "myExt")
    // Registered by IANA; Qaaa ~ Qabx for private use
    .setScript("Latf")
    // Custom key-value pairs.
    .setUnicodeLocaleKeyword("lK", "local-value")
    // Registered by IANA
    .setVariant("1994")
    .build()
    .toLanguageTag()
```

Example 3.1. Creating Locale with Locale.Builder

Beware of `Locale.toString()`

Especially in context of CoreMedia Multi-Site you should not rely on the representation of `Locale.toString()`, which, at first glance seems to be a language tag, just using underscores such as `en_US`. In contrast to IETF BCP 47 language tags this representation has no strict specification and as such cannot reliably be parsed from String representation back to a valid `Locale`. Use `#toLanguageTag` instead.



3.2.2 Adding Locales to CoreMedia Studio

For consistent language tags within *CoreMedia Content Cloud, Studio* configures locales available to editors in a content item typically located at `/Settings/Options/Settings/LocaleSettings`.

As a site manager or administrator it is your task to define those locales in this content item which may be chosen by your editors and especially when deriving new sites.

To define such a locale, you have to select a valid IETF BCP 47 language tag (see [RFC 5646](#) for details). In general this is a language subtag such as `en`, `fr` or `de` along with a country subtag such as `US`, `GB` or `CH` separated by a dash. Some examples are:

- `en-US` – *English (United States)*
- `en-GB` – *English (United Kingdom)*
- `fr-FR` – *French (France)*
- `fr-CH` – *French (Switzerland)*
- `de-DE` – *German (Germany)*

3.2.3 Example: The Global Site Locale

Guess, you are about to create a global site, which is supposed to serve as an abstract site (see [Abstract Sites \[18\]](#)). You want to have a nice, distinguishable name for this site, displayed in *CoreMedia Studio*, but you do not want to collide with any regular locales within your site hierarchy.

The World Region 001

As already stated in the [Introduction](#), a possible alternative to approaches mentioned here about a *Global* locale, you may as well consider using existing registered region subtags, such as `001`, to represent the region *World*.



Your first guess for a nice representation within *Studio* is a string like *English (Global)*. As stated in [Section 3.2.1, “Defining Valid Language Tags” \[22\]](#) there is no option to directly customize the display name of a locale. Instead, it is time to get creative with means how to build locales matching your needs for a proper display name.

Pragmatic: Let's first come to the pragmatic approach. Knowing that most of the time it is the country or region that is displayed in parentheses, you will soon identify that there is no such country available. But you will get the same result regarding the desired display name *English (Global)* by using variants. In general, you should stick to those variants registered at IANA. But: Java does not validate against registered variants. So, you may use any variant you like if it is well-formed. The solution here is the following valid language tag: `en-Global`. It will create a locale with language English, with no country/region set, and with a variant `Global`.

Compliant: If you are eager to stick to the specification, you may use the Unicode Locale Keyword extension for your custom locale. The following language tag is compliant with the specification and will create a locale with language English, with no country/region set, and with a Unicode Locale Keyword extension for key `tp` (for: *type*) and value `global`: `en-u-tp-Global`.

When querying the display name for the language tag `en-u-tp-Global`, you will get: *English (tp: global)*.

To experiment with more viable options, also directly validating if you have a valid language tag, you may want to look at [section "Tip: Design Help Via Java's Locale" \[25\]](#).

Common Locale Data Repository (CLDR)

Since Java 9 the JDK uses the [Common Locale Data Repository \(CLDR\)](#) for locale data. [JEP 252 \(Use CLDR Locale Data by Default\)](#) introduced this resource and emphasized, that this approach may also trigger changes to the display names of locales by revisions of the CLDR locale data in future JDK releases.



4. Multi-Site Challenges

This chapter helps you to get an overview of editorial actions that may harm the Multi-Site processing. It is recommended not to perform these actions, as failures especially in deeply nested site hierarchies may easily spread and expand.

While CoreMedia Multi-Site provides much automation to ease managing localized contents, it sometimes cannot hide the complexity behind. This complexity becomes obvious during unexpected editorial actions, for example, where the Multi-Site feature tries to solve conflicts, but may do so with wrong assumptions.

4.1 General Multi-Site Content Integrity Note

CoreMedia Multi-Site expects no manual interaction between sites. If you stick to this rule, you should have no problems using the Multi-Site feature. Most problems arise when you do not use workflows to transfer contents from one site to another.

Exceptions exist especially where Multi-Site does not yet provide support or cannot provide help. Most prominent are:

- propagation of deleted contents,
- transfer contents between siblings sites, and
- transfer content from derived to master sites.

4.2 Concurrent Editing Feature

CoreMedia Multi-Site is designed to allow concurrent editing, while localization workflows are in progress. Having this, it especially allows editing master as well as derived contents, while an external translation agency takes care of translating them.

Nevertheless, concurrent editing may conflict with your recent changes. Because of this, *CoreMedia Studio* will show a hint in a nagbar, if your changes may collide with translation results. The nagbar will be shown for both roles of a content within a translation workflow: as master or derived. The relevance for the nagbar is different according to the roles, though:

As editor of a master content item, you should not start a new localization workflow, if another workflow for this master and the same target locale is already running. While sometimes intended, it may result in unexpected translation results since, for example, the workflows may not be handled in the expected order.

As editor of a derived content, you may expect translatable properties to be overwritten soon, when the localization workflow, for example, uploads the translation results. Depending on the robustness of the localization workflow, it may also fail (in other words: escalate), when you have checked out the content while the workflow tries to apply the localization results.

Long story short: If the nagbar appears for a given content, ensure you are aware of your editorial processes and take actions to the content with care.

4.3 Deriving a Site

When a new locale (language, region, or country) should be introduced, all content items and folders of an existing site are copied to a new site folder by the *Derive a new localized site* feature in CoreMedia Studio. Like in localization workflows, this process may fail or produce unexpected results, if contents in the master site show errors or get changed whilst deriving the new site. Contents in error may also impair localization workflows targeting your new derived site, later on. This section guides you in preparing your site derivation so that it will succeed.

Short Recommended Checklist

This summarizes the details as given below:

- Ensure you have enough privileges creating the new site.
- Fix content issues in master site to derive from, especially those related to localization.
- Ensure, that all contents in master site are checked in.
- Stop any editorial work (manual or automatic) on master site.



These are the things to check before you try to derive a site. For clarification, *master site* here refers to the site you derive from, not (necessarily) the root master site of your web presence.

- You have *read* rights on *all* contents in the master site. Check with your administrator when in doubt.
- You have *edit* rights on *all* contents in the to-be-derived site. The newly derived site's folder will be a sibling of the master site, using the country and language name. Example: if you derive a site for *Spanish (Argentina)* on your web presence *CoolBrand*, a new folder `/Sites/CoolBrand/Argentina/Spanish` will be created (depending on settings in your site model). Your CoreMedia Studio user must have *edit* rights for contents in this folder. Check with your administrator when in doubt.
- Ideally, all editorial work on the master site is stopped for the time of content check and actual site derivation. If possible, define an official "maintenance window" for this task. Instruct editors to finish all workflows, to check in all contents, and to make sure that no problematic links are set prior to the beginning of the maintenance window (see following paragraphs for details).
- All workflows targeting the master site should be finished. Example: you want to derive a new site from *Spanish (Spain)*. All workflows with target site

Multi-Site Challenges | Deriving a Site

Spanish (Spain) should thus be finished before you try to derive your new site.

- Likewise, no automated content imports should happen during the derivation. Stop all corresponding processes and inform external partners not to modify contents of your master site.
- All contents in the master site should be checked in. Contents *must* at least have been checked in once. The easiest way to ensure this is to check them all in. Use the CoreMedia Studio library search with appropriate filters to see all checked-out content items in your master site (search filter *Status / In Production / Checked out by...*).
- No contents in the master site show errors related to Multi-Site. Use CoreMedia Studio library search with appropriate filters to see all contents with errors (search filter *Issues / Category: Localization*). It may take some time for issues to appear in the search, so wait a few minutes after editorial work has been stopped. Check and fix all errors you see, especially
 - links to deleted content items,
 - cross-site links, and
 - wrong locale settings.

You should in addition use the **cm** tool *validate-multisite*, as described in [Section 3.13.1.1, "Validate Multi-Site"](#) in *Content Server Manual* to check for broken contents in the master site.

4.4 CleanInTranslation Challenges

What is CleanInTranslation? Reading the title of this section, you may not even know about `CleanInTranslation`. It can be summarized as the *garbage collector* for localization workflows: If a localization workflow aborted abnormally (escalated, for example), it will leave traces within contents, which mark them as being *in translation*.

Technically, it runs as scheduled task within the workflow server. See [Section 5.7.2.3, “Configuration and Customization”](#) in *Blueprint Developer Manual* for some configuration options for this scheduled task.

Effects not running CleanInTranslation: Not performing such garbage collection, may provide irrelevant *in translation* state reports in *CoreMedia Studio* and it will block affected versions of a content item being in translation from being cleaned up by processes such as `DestroyIntermediateVersions` or tools like *cleanversions* (see [Section “Clean Versions”](#) in *Content Server Manual*). This is relevant also, when `CleanInTranslation` has issues while performing its cleanup tasks.

These issues, which block cleanup, are, what this section is about. It tells you:

- [How to detect issues? \[33\]](#)
- [How to dump workflow processes? \[34\]](#)
- [How do mostly normal operation looks like? \[36\]](#)
- [How to analyze issues? \[37\]](#)
- [How to fix issues? \[38\]](#)

How to detect issues?

If there are no issues with your workflow and content states, the scheduled task `CleanInTranslation` will regularly output reports as shown in [Example 4.1, “Standard Activity in Workflow Server Log” \[33\]](#).

```
[INFO] com.coremedia.translate.workflow.CleanInTranslation [] -
Merge-Version Cleanup: Started...
[INFO] com.coremedia.translate.workflow.CleanInTranslation [] -
Merge-Version Cleanup: Done within 11.24 ms,
cleaned up merge versions: 0.
```

Example 4.1. Standard Activity in Workflow Server Log

If things start going wrong, you will find, in addition to the standard operation report, warnings such as shown in [Example 4.2, “Warning in Workflow Server Log” \[34\]](#) in Workflow Server Log.

```
[WARN] com.coremedia.translate.workflow.CleanInTranslation [] -  
Process[coremedia:///cap/process/2288]:  
Process is in an invalid state regarding the stored derived contents and  
their master content objects.  
See debug log for more details.  
Please consider aborting the process or repairing the content.  
Tools which might help you: `cm validate-multisite`, `cm processes`,  
`cm dump`.
```

Example 4.2. Warning in Workflow Server Log

Having this, it is about time to take action, as the warnings will not disappear without intervention.

How to dump workflow processes?

At some point in time during issue analysis, it may be required to get the details of an affected localization workflow process. For example, if you want to analyze a reported process without increasing the log level to `DEBUG` or to repair a content state to the state prior to starting the localization workflow.

This section will give you some hints, how you may analyze such a process reported to have issues.

```
cm dump -u admin coremedia:///cap/process/2288 --blob text
```

Example 4.3. Dump Localization Process

```
process: coremedia:///cap/process/2288  
definition: Translation (coremedia:///cap/processdefinition/8)  
properties:  
  autoMergeConflicts=[]  
  createdContents=[  
    Content[coremedia:///cap/content/124]  
  ]  
  derivedContents=[  
    Content[coremedia:///cap/content/124],  
    Content[coremedia:///cap/content/144]  
  ]  
  masterContentObjects=[  
    Version[coremedia:///cap/version/42/1],  
    Version[coremedia:///cap/version/44/1]  
  ]  
  premarularConfigData=Blob[content type=application/json, size=137],  
  Content:  
  ...  
  {  
    "coremedia:///cap/content/124":  
    "coremedia:///cap/version/42/1",  
    "coremedia:///cap/content/144":  
    "coremedia:///cap/version/44/1"  
  }  
  ...  
  rollbackVersions=[  
    Version[coremedia:///cap/version/144/1],
```

```
[...]
```

Example 4.4. Dump Output

When invoking `cm dump` similar to [Example 4.3, “Dump Localization Process” \[34\]](#) you will get an output similar to [Example 4.4, “Dump Output” \[34\]](#). The output is shortened here, to the most important details required for analysis and fixing `CleanInTranslation` issues. Other – possibly valuable information – include involved users or timestamps, telling about the age of the workflow process and its tasks.

Given the example output, it tells you:

definition: The workflow process definition, which is affected here. You may require analyzing the implementation of this process, if things go wrong in general.

Property `autoMergeConflicts`: This property tells about possible conflicts when merging, for example, changes in link lists. Such conflicts sometimes trigger actions by editors, which in the end may cause issues with `CleanInTranslation`. Thus, in this context, it is sometimes helpful to understand possible causes of invalid states.

Property `createdContents`: These contents were created in derived site, as they did not exist before, but are required to receive the localization result. If aborting an affected localization workflow, you may want to delete these contents, unless they are meanwhile used otherwise, like referenced in links.

Property `derivedContents`: Lists all derived contents that should receive localization results.

Property `masterContentObjects`: Lists all master contents that either got added explicitly or implicitly (as dependent contents) to the localization workflow. If issues are related to these master contents, you may sometimes want to call `cm dump` for these, too, to understand their history.

Property `premarlarConfigData`: This is an artificial workflow variable, required in *CoreMedia Studio* to display localization workflows. Nevertheless, its contents may help you to spare some calls to `cm dump`. This is because it is the only place, where you directly see the relation from derived contents to their corresponding master contents. Thus, in the example you see, that the derived content with ID 124 is derived from master content having ID 42, and derived content with ID 144 is derived from master content with ID 44.

Property `rollbackVersions`: During the localization process, previously existing derived contents will be edited. Possibly even at different stages, with a bunch of versions created. This property stores the state of the derived content before the localization process started. It will be used to restore the state, when

a rollback is chosen during the workflow execution. In context of `CleanInTranslation` issues, these versions may be important if you want to manually restore a state prior to starting the affected localization workflow. Nevertheless, you should take care to analyze, if editors did not meanwhile create additional versions prior to reverting to the version given here.

Note, that the actual dump may look different for custom localization workflows. But you should be able to locate comparable information inside these custom workflows.

How do mostly normal operation looks like?

```
[INFO] Merge-Version Cleanup: Started...
[DEBUG] Merge-Version Cleanup: Stage 1 consumed 39.81 ms.
[DEBUG] Blacklisted derived contents: []
[DEBUG] Processed 0 derived contents.
[DEBUG] Merge-Version Cleanup: Stage 2 consumed 56.28 µs.
[DEBUG] Merge-Version Cleanup: Stage 3 consumed 17.35 µs.
[INFO] Merge-Version Cleanup: Done within 40.45 ms,
       cleaned up merge versions: 0.
```

Example 4.5. Standard Activity in Workflow Server Log on Debug Level

For a well-informed start analyzing possible issues, it may be good following the advice increasing the log level to `DEBUG` for `logger.com.coremedia.translation.workflow.CleanInTranslation` as suggested in the log message. [Example 4.5, “Standard Activity in Workflow Server Log on Debug Level” \[36\]](#) shows the messages provided in addition to the default ones for normal operation (logger name stripped for better readability). When localization workflows are active, only the number of processed derived contents will increase.

```
[INFO] Merge-Version Cleanup: Started...
[...]
[DEBUG] Processed 1 derived contents.
[DEBUG] Merge-Version Cleanup: Stage 2 consumed 81.39 µs.
[DEBUG]
       Registered merge version Version[coremedia:///cap/version/42/1]
       on content Content[coremedia:///cap/content/124]
       for possible cleanup.
[DEBUG] Merge-Version Cleanup: Stage 3 consumed 3.965 ms.
[DEBUG] Merge-Versions scheduled for possible clean-up on next iteration(s):
1
[INFO] Merge-Version Cleanup: Done within 24.60 ms,
       cleaned up merge versions: 0.
```

Example 4.6. Content Cleanup Preparation

```
[INFO] Merge-Version Cleanup: Started...
[...]
[DEBUG] Processed 1 derived contents.
[DEBUG] Merge-Version Cleanup: Stage 2 consumed 85.30 µs.
[DEBUG]
       Registered merge version Version[coremedia:///cap/version/42/1]
```

```
on content Content[coremedia:///cap/content/124]
for possible cleanup.
[INFO]
  Cleaning up merge version Version[coremedia:///cap/version/42/1]
  on content Content[coremedia:///cap/content/124]
[DEBUG] Merge-Version Cleanup: Stage 3 consumed 6.216 ms.
[INFO] Merge-Version Cleanup: Done within 27.01 ms,
  cleaned up merge versions: 1.
```

Example 4.7. Content Cleanup Execution

Another still normal operation is shown in [Example 4.6, “Content Cleanup Preparation”](#) [36] and [Example 4.7, “Content Cleanup Execution”](#) [36]: It signals, that a localization workflow got aborted (manually or by escalation) and that cleanup is scheduled for next iteration of the scheduled task. The message *Registered merge version* tells in the given example, that version number 1 (one) of master content with ID 42 will get its *in translation* state cleaned. It got locked, because it was meant to be localized towards derived content with ID 124.

In the next iteration, this version gets cleaned up and any false positive *in translation* state got removed.

Side Note on Cleanup Stages: Due to some asynchronous behavior between content and workflow server, cleanup should not be performed immediately. That is why cleanup with default configuration is only performed, when a given master version is recognized at least twice being in false positive *in translation* state. Note, that because of this, it may also happen, that a previously detected version to cleanup is vetoed in next iteration. This is normal operation and does not need extra awareness.

How to analyze issues?

First, a note in advance: Analyzing issues is not required to get rid of corresponding warnings from `CleanInTranslation`. But it is strongly recommended, as ignored issues may leave contents in an unexpected state hard to recover from later. The affected process is the only reference, which will tell you about the original states, so that you should not abort/destroy it prior to having analyzed its state.

In the end the standard recommended process is:

- analyze issues,
- fix affected contents,
- and most likely destroy affected process eventually.

```
[DEBUG] Issues found while retrieving derived contents and merge versions
from process coremedia:///cap/process/2288
via strategy DefaultTranslationWorkflowDerivedContentsStrategy: issues (2)
=
```

```
[  
  Issue[type = ...],  
  Issue[type = ...]  
]
```

Example 4.8. General Issues Report

As soon as the logger is set to `DEBUG`, `CleanInTranslation` will report issues similar to [Example 4.8, “General Issues Report” \[37\]](#). Each of these issues should be analyzed and addressed accordingly prior to aborting/destroying the corresponding process.

Note, that in previous CMS versions, the output differed slightly. But it provided the same or similar information.

How to fix issues?

In the following sections we will now address the various issue types, what they tell you and how to address or fix those issues.

Workflows will escalate: Any workflow affected by these issues, is not operational anymore. That is, trying to continue proceeding with the given localization workflow will trigger an escalation at some point. That is, why most of these workflows need to be aborted (destroyed) once you have taken care of the issues, unless there is a way to restore the original state.

The issue types mentioned are:

- [DERIVED_DELETED \[39\]](#)
- [DERIVED_DESTROYED \[40\]](#)
- [DERIVED_MASTER_MISMATCH \[40\]](#)
- [DERIVED_NO_MASTER \[41\]](#)
- [MASTER_DESTROYED \[41\]](#)

The list of possible issue codes represents those, assumed to be relevant for analyzing `CleanInTranslation` failures. If you experience other issues codes, please contact our support team, so that they may help you.



Recommended Tools

The following sections describe command-line tools, that are recommended for analyzing and possibly fixing issues:

- `cm destroy`: [Section 3.13.3.3, “Destroy”](#) in *Content Server Manual*
Used to get rid, for example, of affected localization workflow processes.
- `cm dump`: [Section 3.13.1.1, “Dump”](#) in *Content Server Manual*
Used to analyze affected workflow processes and contents. Note, that it is recommended providing the full ID of involved objects such as `coremedia:///cap/process/2288` or `coremedia:///cap/content/124`. Numeric IDs will just reference contents.
- `cm processes`: [Section 3.5.7, “Processes”](#) in *Workflow Manual*
May be required, to see, if contents are involved in any other workflow processes.
- `cm validate-multisite`: [Section 3.13.1.1, “Validate Multi-Site”](#) in *Content Server Manual*
Always a good tool, to analyze the overall state before and after addressing issues. As alternative, most of the issues reported here are also reported as validation issues within *CoreMedia Studio*. Use the corresponding library filter for locating them.

DERIVED_DELETED

```
Issue[
  type = DERIVED_DELETED,
  master = coremedia:///cap/version/42/1,
  derived = coremedia:///cap/content/142,
  message =
    "Derived content is deleted: Content[coremedia:///cap/content/142]"
]
```

Example 4.9. Issue Output for `DERIVED_DELETED`

The reported issue given in [Example 4.9, “Issue Output for `DERIVED\_DELETED`” \[39\]](#) signals, that the derived content that is meant to receive a localization result got deleted meanwhile.

How to fix? A possible option is restoring the affected content from recycle bin. Otherwise, the localization workflow cannot continue.

If restoring is not an option, you most likely want to try to restore the original content state prior to starting the localization workflow. For details how to make

use of the affected localization workflow, see [section “How to dump workflow processes?”](#) [34].

Note on older CMS versions: In previous versions, the `master` was reported to be always unset (`null`). If this is the case, you may need to `cm dump` the derived content instead, to see its related master content.

DERIVED_DESTROYED

```
Issue[
  type = DERIVED_DESTROYED,
  master = <unset>,
  derived = coremedia:///cap/content/13274,
  message =
    "Derived content is destroyed: Content[coremedia:///cap/content/13274]"
]
```

Example 4.10. Issue Output for `DERIVED_DESTROYED`

The reported issue given in [Example 4.10, “Issue Output for `DERIVED\_DESTROYED`”](#) [40] signals, that the derived content that is meant to receive a localization result got destroyed meanwhile. Note, that the master cannot be provided, as the content got destroyed. Instead, you may want to analyze the workflow details to find the related master content.

How to fix? Unless you have some backup, there is no way to restore this derived content. Thus, you most likely need to revert to the state prior to starting the affected localization process. For details how to do that, see [section “How to dump workflow processes?”](#) [34]. Another alternative is, to review all contents related to the workflow and try to push them towards some useful state. For older localization workflows, it may even be, that editors meanwhile manually fixed the related contents, so that there is nothing to do.

DERIVED_MASTER_MISMATCH

```
Issue[
  type = DERIVED_MASTER_MISMATCH,
  master = coremedia:///cap/content/42,
  derived = coremedia:///cap/content/124,
  message =
    "Unable to determine master version for derived content:
    /Sites/Chef Corp./de-DE/Editorial/Document
    <Content[coremedia:///cap/content/124]>.
    Currently set master Content[coremedia:///cap/content/42] is not
    represented within master content objects as expected."
]
```

Example 4.11. Issue Output for `DERIVED_MASTER_MISMATCH`

The reported issue given in [Example 4.11, “Issue Output for `DERIVED\_MASTER\_MISMATCH`”](#) [40] signals, that there is no corresponding master referenced in the variable `masterContentObjects` of the corresponding workflow. The given value for `master` is the master as referenced by the derived content.

A possible reason may be, that the master got destroyed, which would raise another issue: `MASTER_DESTROYED` [41]. Other possible options are, that the workflow got started with an unexpected state regarding its `derivedContents` and `masterContentObjects` variable or that the master link within the derived content got changed meanwhile.

How to fix? Assuming a relation to `MASTER_DESTROYED` [41], the approach is similar or the same: You need to understand the state of the derived content and repair it accordingly. This may mean to delete it or to adjust master and master version property.

If the workflow got started with an irregular state, or the master link got changed meanwhile, you may instead want to analyze the workflow and `cm dump` the process to analyze its properties. See section “How to dump workflow processes?” [34] for details. The result may help to repair the content or to understand possible issues of the affected localization process definition.

DERIVED_NO_MASTER

```
Issue[
  type = DERIVED_NO_MASTER,
  master = <unset>,
  derived = coremedia:///cap/content/124,
  message =
    "Required master missing for derived content:
    /Sites/Chef Corp./Germany/German/Example
    <Content[coremedia:///cap/content/124]>"
]
```

Example 4.12. Issue Output for `DERIVED_NO_MASTER`

The reported issue given in Example 4.12, “Issue Output for `DERIVED_NO_MASTER`” [41] signals, that the derived content meanwhile has no corresponding master property set anymore.

How to fix? Most likely, some previous version of the derived content still references the master. Thus, you may want to restore the master link. Afterwards, the localization workflow can be completed. Prior to that, you may want to understand why the master link got deleted.

If the version that references the master is not available anymore (for example, due to version cleanup), you may instead have a look at the `cm dump` results to restore the master link according to those data. For details see section “How to dump workflow processes?” [34].

MASTER_DESTROYED

```
Issue[
  type = MASTER_DESTROYED,
  master = coremedia:///cap/version/42/1,
  derived = <unset>,
  message =
```

```
"Master content referenced by process is destroyed:  
Version[coremedia:///cap/version/42/1]."  
]
```

Example 4.13. Issue Output for MASTER_DESTROYED

The reported issue given in [Example 4.13](#), “Issue Output for MASTER_DESTROYED” [41] signals, that the version or content referenced in the variable `masterContentObjects` of the corresponding workflow got destroyed.

The issue generally comes with an additional issue [DERIVED_MASTER_MISMATCH](#) [40], as a missing master also means, that the corresponding master for a given derived content cannot be found.

How to fix? Having this issue means, that there is most likely some orphaned derived content, now linking to this destroyed master. As stated, this orphaned derived content should be referenced in an issue of type `DERIVED_MASTER_MISMATCH`. If it is not mentioned, then the localization workflow got started with an unexpected set of master and derived content objects, which, at least here, are not related to another.

Ignoring the irregular state, the easiest way to proceed is to find the corresponding issue of type `DERIVED_MASTER_MISMATCH` which references master with ID 42 in the given example.

Now everything, which is left to do, is to decide, how to deal with the orphaned derived content. Possible options are to remove it or at least repair the master link and master version property. For example, if it should have no master anymore, just clear both entries. Any other options require understanding the history of this derived content and its relation to other contents, like links referencing the derived content.

4.5 Referrer Relations in Localization

Some content types form tree hierarchies through parent-child relationships. The most prominent example is the navigation structure: `CMChannel` content items reference child channels, forming the sites page tree.

When translating a content item that belongs to such a tree, the parent content items may not yet exist in the target site. Without the parents, the translated content item ends up as an orphan structurally disconnected from the rest of the tree.

Automatic Referrer Transfer Augmentation

CoreMedia Multi-Site may consider referencing parent content items when computing the translation set based on the provided configuration. If configured and enabled for navigations channels, when localizing a navigation channel, all ancestor channels up to the root channel are considered automatically at any depth.

Include Updated Dependent Content

By default, only parents are considered that do not exist yet in the target site.

For parents that already exist in the derived site, editors should activate the option to include updated dependent content when starting the localization workflow. This ensures that structural link-list changes are propagated to derived sites even when the parent content item already exists.



Built-In Relations

Several referrer relations are pre-configured and enabled by default:

navigation	Covers <code>CMNavigation</code> (and its subtypes such as <code>CMChannel</code>). The parent-child relationship is defined by the <code>children</code> link-list descriptor.
navigation-root	Includes <code>CMSite</code> documents that reference a root navigation channel via the <code>root</code> property.
taxonomy	Covers <code>CMTaxonomy</code> (and its subtypes). The parent-child relationship is also defined by the <code>children</code> link-list descriptor.

taxonomy-root Includes `CMSSettings` documents that reference top-level taxonomies via the `settings` struct property.

Custom Referrer Relations

If your content type model includes custom content types with structural parent-child relationships, you can register additional referrer relations. For example, given a custom content type `AcmeCategory` with a `subcategories` link-list descriptor:

```
multisite.transfer.relations.referrer.acme-category.content-type=AcmeCategory
multisite.transfer.relations.referrer.acme-category.source-property=subcategories
```

If your custom content type replaces one of the built-in types (for example, a custom taxonomy type that replaces `CMTaxonomy`), you can override just the content type the other properties are preserved from the defaults:

```
multisite.transfer.relations.referrer.taxonomy.content-type=AcmeTaxonomy
```

Configuration

All properties are prefixed with `multisite.transfer.relations`. The full reference is available in [Section 3.6.2, “Shared Multi-Site Configuration”](#) in *Deployment Manual* as well as in [Section 3.4.3, “Shared Multi-Site Configuration”](#) in *Deployment Manual*.

To disable a specific relation:

```
multisite.transfer.relations.referrer.navigation.enabled=false
```

To disable referrer relation augmentation entirely:

```
multisite.transfer.relations.enabled=false
```

Keep Studio Server and Workflow Server in Sync

Referrer relation augmentation runs on two applications: *CoreMedia Studio* (Start Localization Workflow window) and the Workflow Server (actual augmentation during the workflow). Both must use the same configuration. If the configuration diverges, editors may see a set of additional content items that does not match the content items actually added by the workflow.



4.6 General Multi-Site Fixes

Most issues with CoreMedia Multi-Site can be fixed by adjusting the localization properties after unexpected actions such as copy and paste of contents between sites. These are the expected states of the properties you should know:

- Locale:** This is expected to have the same locale as the site the content is in.
- Master:** This is expected to be either empty, if it is a content independent of the master site, or it should link to its variant in the master site, so that it receives updates from the master site.
- Master Version:** This is expected to contain any valid version number of the master. A negative value indicates that the content has been derived from this (positive) master version but has still to be translated. If the master link is empty, this version should be empty, too.

In rare cases, the field can contain the value 0 (zero). This is not a real version. It indicates that something went wrong. See *Invalid Master Version* in [Table 4.1, “List of Multi-Site Challenges” \[46\]](#) for details.

4.7 Overview of Possible Pitfalls

Table 4.1, “List of Multi-Site Challenges” [46] is grouped by these keywords:

Symp-toms	Lists <i>possible</i> symptoms, you may observe. The symptoms are not necessarily complete, but are the most prominent symptoms you may observe.
Causes	The possible causes for the observed symptoms. Again, there may be additional or other causes, but the ones listed are those known to occur sometimes.
Effects	Effects, if ignoring this possibly erroneous state. The severity of these effects may be minimal, thus, ignoring a given state may always be an option.
Fixes	How to resolve the issues. Sometimes, there are several approaches, which you may consider, depending on your perceived severity of the issue.
Prevent	What is recommended to prevent such issues. Most of the time, it lists best practices when dealing with multi-site content.

Validate Multisite

Most of the symptoms below can be monitored with the **cm** tool *validate-multisite*, as described in [Section 3.13.1.11, “Validate Multi-Site”](#) in *Content Server Manual*.



Cross-Site Links	
Symptoms	Validation Issue: Most likely you observed this issue when opening the affected content in <i>CoreMedia Studio</i>. It will tell you about links that refer to other sites, ignoring the master link that is meant to link to other sites. Confused Site Visitors: Your site visitors will experience switching between languages accidentally just by navigating through your site.
Causes	Manually Copy Contents Between Sites: Despite the most obvious reason, that you may have manually added links from other sites, the more likely reason may be copying contents between sites. If you do so, there is no feature, which will map contained links, as it will be done in translation and synchronization workflows. Thus, instead of links pointing to the site you copied to, the links will still refer to the source site you copied from.

	Manually Copy Richtext Between Sites: A similar, may be less obvious cause, is copying Richtext contents from one site to another. They may contain embedded images as well as internal links. On copy & paste these images/links will still link to the originating site.
Effects	Workflows Not Updating Links: From a translation or synchronization workflow perspective, these links most likely will be considered as independent links in a derived site, thus, as if they have been explicitly added in the derived site. These links will never get updated by translation or synchronization workflows, they may even cause conflicts when trying to merge changes to the affected link lists. This effect will only occur, if you have explicitly set a master-link. Note, that in case of copy & paste master-links and master-versions will be removed, if set. Delivery Issues: Depending on your delivery configuration, you may experience either broken links or (more likely) your visitors stumbling from one language to another without being aware of it.
Fixes	Remap Links Manually: If recreating the content from master is not an option, you may manually remap the links to link to the same contents in the actual site. Note, that this applies to link lists, to links and embedded images in CoreMedia Richtext just as links set in Struct properties.
Prevent	Use Workflows: Instead of manually copying contents from one site to another, use workflows such as translation or synchronization workflow (whichever applies). Workflows will take care of mapping the links to the corresponding sibling in the target site. In addition to that they will also create possibly missing link targets in derived sites. Remap Links Manually: Just as stated above, if you copied contents or property values such as CoreMedia Richtext from one site to another, you have to manually remap the links accordingly. Note, that this may not be feasible in <i>CoreMedia Studio</i>, as not necessarily all links are accessible within <i>CoreMedia Studio</i>. If in doubt, please ask your administrators to check the content.
Deleted Master	
Symptoms	Validation Issue Deleted Master: In <i>CoreMedia Studio</i> you will see an issue, that the master property refers to a non-existing master.
Causes	Deleted Master: Most likely, the master content got deleted. Unfortunately, there is not yet a way to propagate deletion of contents from one site to another. Thus, for a vivid site, it is likely that you will stumble across this issue.

Effects	No more updates: The content affected by deleted master will not receive any updates anymore. Most likely, as soon as all referrers have been updated, there will be no more content items linking to this content, and thus, you will have some content in your site unreachable by site visitors. Despite this, the state is rather harmless and is similar to a content intentionally just existing in a derived site, but not in the master site.
Fixes	Delete Derived Content: You may want to delete the derived content, just as the master content got deleted. Prior to that, you may want to run a translation or synchronization workflow for all contents linking to this content, as it is expected that any links will be automatically removed by this process. Remove Master Link: Of course, if you want to keep this content, you can simply remove the master link. It is recommended to clear the master version property as well. Restore Master: Always an option is to restore the master.
Prevent	As there is now no way to propagate deletion of contents, there is no way to prevent this state, other than establishing a communication of site administrators of the master site to the local site administrators.
Duplicated Content Items	
Symptoms	Some Document (1): In your derived site you may observe, that just next to your content <code>Some Document</code> another content named <code>Some Document (1)</code> exists.
Causes	Create With Same Name: You may have created a content with the same name, as it now exists in master site. When transferring the master content to your derived site, name conflicts are automatically resolved and the content from master will get a counter added to its name. Manually Copy Between Sites: A typical pattern, which has been observed is that contents are copied manually between sites, instead of using the appropriate workflows.
Effects	Ignoring state may be an option: From the repository perspective you may safely ignore this state, as the name of the content is irrelevant for the Multi-Site feature to work. The relationship is always derived from the linked master content. But it may indicate, that a content receives translations, which is not even included in your site navigation. If you continue working with these contents, only the content that links to the master via master-link will receive updates.

Fixes	Fix Relation: If you expect that both content items are identical, you may need to resolve the conflict. Decide, which content item matches your desired content item best (most likely the one you already had in your site before).
Prevent	Use Workflows: Instead of manually copying contents from one site to another, use workflows such as translation or synchronization workflow (whichever applies). Note, that this is only possible from master site to a directly derived site. Adjust Localization Properties: If you need to copy a content to your derived site, ensure to adjust the localization properties. More specifically: Locale: Set the locale to the same locale as the site your content item is in. Master: Adjust the master-link, so that it links to the same content item in the master site. Master Version: Adjust the master version. Most likely, this is the current version number of the master (you will find it on System tab). If you want to signal, that this content item is not translated yet, you may negate this version number. Thus, if you have copied version 5, but you want to mark the content item as <i>not translated yet</i>, set the version number to -5.

Must Not Duplicate Master Link

If you adapted the master-link but kept the duplicated content, two contents will refer to the master. Subsequent translation and synchronization processes will resolve this ambiguity by choosing a random target content.

Thus, always ensure that a master content is always only referenced once from a given derived site.

Invalid Master Version	
Symptoms	Changes of Links Not Propagated: If you trigger a translation or synchronization workflow, you experience, that links changed in master do not make it to the derived content.
Causes	Unset Master Version: The master version property may be empty, that is unset. Technically, the field is set to <code>null</code> . This triggers the same behavior as:

	Invalid Master Version: The master version may denote a version, which does not exist or does not exist anymore for several reasons. Negative values are valid, if their absolute value matches an existing version. The actual cause of such states is unknown. It may be a special case of copy & paste between sites, but more likely any other tools that caused this invalid state. One example may be, that a version got destroyed by cm tools.
Effects	Properties Not Updated As Expected: For any property that is not meant to be translated, the expectation is, that it gets automatically updated from master. This feature is called auto-merge. This feature relies on the ability to determine, which properties have been changed from previous translation or synchronization to now. If the previous state of properties of the master cannot be determined because the master version is unset or not existing the defensive behavior of CoreMedia Multi-Site will assume no changes to apply.
Fixes	Set Master Version: If you are lucky, you may guess an appropriate master version that matches the state of your derived content. If in doubt, set it to the current master version and review non-translatable properties such as links in differencing view. If you have multiple contents in this state, consider setting it to the negative current master version. This way, you will have a kind of to-do-list, as these contents will be listed as not translated yet. Finish Workflow: Finishing a translation or synchronization workflow will adjust the version property automatically. This may be the easiest option to take, but you may miss contents you should have reviewed.
Prevent	Handle With Care: As the actual cause is unknown, there is not much more to say than to handle the master version property with care. It contains important information, which, if invalid, may harm your translation and synchronization processes. Any tooling and processes provided by CoreMedia respect these versioned references, so that they are not automatically cleaned up. The only difference is a tool called cm destroy, which is especially meant for administrative emergency interventions.
Locale Mismatch	
Symptoms	Validation Issue: Most likely you observed this issue when opening the affected content in <i>CoreMedia Studio</i>. It will tell you about a mismatch between the site's locale and the locale set for the content.

Causes	Intended Behavior: This may be intended behavior, that is, you may want to provide a content with a different locale in the same site. Note, that due to current limitations of CoreMedia Multi-Site it is not recommended to use locales different to the site locale within a given site. For details see Effects. Manually Copy Between Sites: A typical pattern, which has been observed is, that contents are copied manually between sites, instead of using the appropriate workflows. If doing so, the locale will not be updated accordingly.
Effects	Ignoring state may be an option: Using different locales within a site may be a valid option, if the following effects are not relevant to you. Accidental Locale Clash on Derive Site: The effect is best described using an example: Think of a master site having locale <code>en-US</code>. This site contains a content D with German locale <code>de-DE</code>. If you derive a new site with German locale, the derived content D' will be perceived as having the site's locale. In the last step, you will derive a new site with Dutch locale <code>nl-NL</code> from the German site. As a result content D" in the Dutch site will now be adapted to locale <code>n-NL</code> which may not be desired. <i>Figure 4.1. Conflict Site vs. Content Locale</i> In other words: If the site containing content D is a so-called leaf-site, thus, having no derived sites, and will never have one, there is no real issue having this content, unless you do translation via XLIFF: Wrong Locale on Translation: Having a content D with locale <code>de-DE</code> in a site having locale <code>en-US</code> will result in XLIFF being generated with site-locale <code>en-US</code>. Thus, translation agencies will not be aware of the correct locale. This applies to the target locale as well as to the source locale.
Fixes	Fix Locale: If you copied the content from another site, adjust the locale accordingly. It should be the same as the site's locale.
Prevent	Use Workflows: Instead of manually copying contents from one site to another, use workflows such as translation or synchronization workflow (whichever applies). Note, that this is only possible from master site to a directly derived site. These workflows will automatically adapt the locale.

Adjust Localization Properties: If you need to copy a content to your derived site, ensure to adjust the localization properties. In this case:

Locale: Set the locale to the same locale as the site your content is in.

Removed Master Link On Copy	
Symptoms	You copied a content and now see the hint "No master document has been specified".
Causes	When a content is copied in Studio, its master link is removed intentionally to maintain consistency and prevent further issues with localization.
Effects	If there is a respective content in the master site that should be referenced as master from the current content, you might miss updates if the reference is not added.
Fixes	If there is a respective content in the master site, you should add a link to that using the appropriate version. If the related master content can be identified by the system, as, for example, by the same path relative to its site, you should see a suggestion in the issues for a master content. If there is no respective content in the master site, nothing needs to be done.
Prevent	If you wanted to create a derived content for an existing master content, consider using the respective workflows, for example, translation or synchronization.

Table 4.1. List of Multi-Site Challenges

4.8 Preconditions for Comprehensive Changes

No matter if, for example, changing the site type from translation to synchronization or restructuring the hierarchy of your sites: You should always consider the implications of your changes. For endeavours like these, you can reduce complexity and risks by ensuring some preconditions. This section provides a summary of the most important preconditions to consider before applying such changes.

The preconditions to fulfill are:

- [Complete Pending Workflows](#) [53]
- [Content Items are Checked-In](#) [53]
- [Content Items are Up-to-Date](#) [53]
- [Editorial Work Stopped](#) [54]
- [Multi-Site State is Valid](#) [54]

Complete Pending Workflows

This especially applies to translation workflows, as, by nature, they have a longer lifetime than synchronization workflows:

Ensure, that any localization workflows within your affected sites are completed. If not doing so, you may experience your pending workflows to escalate or to cause inconsistencies in the content items of your sites.

Find Pending Workflows: As you may not see all relevant workflows in *Core-Media Studio*, consider using the command line tool **cm processes**: [Section 3.5.7, "Processes"](#) in *Workflow Manual*.

Content Items are Checked-In

Ensure that all content items in the affected sites are checked in.

Depending on the type of change, you may or will need to adapt content items like adjusting the master version or the `ignoreUpdates` flag. To do so, administrative users can override a checked-out-by-other state by forcing the check-in. It is much more transparent though for editors to take over the check-in of their content items themselves.

Content Items are Up-to-Date

Ensure that all content items within your affected sites are up-to-date in terms of the translation state. If not doing so, subsequent required steps to update

Multi-Site Challenges | Preconditions for Comprehensive Changes

the master-versions of the content items may be more complex or even not manageable at all.

Editorial Work Stopped

While applying the changes, no editorial work should be done within the affected sites. This is especially important, if your applied changes cause mildly irregular states of your sites while applying the changes. This applies to but is not limited to changing the site hierarchy.

Multi-Site State is Valid

Any issues in the affected sites may impair the change process. Make sure your affected sites are all valid.

For a start, you can check for issues within your site by enabling the search filter *Issues* in the library of *CoreMedia Studio*, searching within the category *Localization*.

For more thorough checks, you can also use the command line tool **cm validate-multisite** as described in [Section 3.13.1.11, “Validate Multi-Site”](#) in *Content Server Manual*.

Expect Issues

Due to the complexity of the multi-site feature, it is not uncommon to find issues in the sites. You may find even more issues after applying comprehensive changes within the site hierarchy.

A process of first validating the sites and fixing all relevant reported issues before applying changes will help to identify new issues that occur after the changes. Also, it is good practice to eventually validate the multi-site state after applying the changes, to ensure frictionless editorial experience after the change.



4.9 Changing the Site Type

This section is a first introduction to the more complex topic of changing the hierarchy of your sites (see [Section 4.10, “Changing the Site Hierarchy” \[57\]](#)). It addresses a more lightweight change, namely changing the site type from **sync** to **translation** or vice versa.

Carefully Design Multi-Site Hierarchy

As already mentioned in [Section 3.1, “Designing Site Hierarchies” \[16\]](#), changing the hierarchy of your sites is a challenging task. The same applies to just changing the site type. Make sure you have a clear understanding of the implications of your changes before you start.



4.9.1 The Site Type

There are two possible types of derived sites in a multi-site setup:

- Translation
- Synchronization

To get to know more about these two types of sites, see [Section 3.1, “Translation and Synchronization” \[16\]](#).

Typically, the site type is selected once in the lifetime of a site, which is when deriving the site from the master site.

The type is stored in the settings of your site at:

- Options/Settings/Translation.

The type is determined by the struct named `translationStrategies` and can be either `SYNC` or `LANG_TRANSLATION`, denoted by the name of a corresponding nested struct node. See [Example 4.14, “Translation Settings Struct” \[55\]](#).

```
translationStrategies:  
  # Empty Struct; alternative key: 'SYNC'  
  LANG_TRANSLATION: {}
```

Example 4.14. Translation Settings Struct

4.9.2 Apply Type Change

Fulfill Preconditions

Before you apply the type change, make sure you have fulfilled all preconditions mentioned in [Section 4.8, “Preconditions for Comprehensive Changes” \[53\]](#) and



After having carefully considered the implications of changing the site type, you can proceed with the actual change.

As you may guess from the previous sections and [Example 4.14, “Translation Settings Struct” \[55\]](#) the central point for changing the type of the site is switching the name of the nested struct node of `translationStrategies` from `SYNC` to `LANG_TRANSLATION` or vice versa at `Options/Settings/Translation` within your site.

Implicitly Ignored Content Item

As the type-setting is only relevant for the very site it is located in, it is ignored by any localization workflows. Thus, you can perceive the change as local only and there is no need updating the type-setting in directly derived sites.



4.10 Changing the Site Hierarchy

This section is about the sensitive topic of changing the hierarchy of your sites. Compared to changing the site type (see [Section 4.9, “Changing the Site Type” \[55\]](#)), this endeavor is much more complex and requires a lot of care and consideration. It should be noted, that the original advice given in [Section 3.1, “Designing Site Hierarchies” \[16\]](#) still holds true, which is: “As changing the site hierarchy after initial kickoff is not an easy task, it is recommended to spend some thoughts upfront on the hierarchy in the design phase of your project. ”

It must be emphasized even more than just for a simple change (simple, compared to this endeavor) of the site type (from translation to synchronization or vice versa), **all preconditions must be fulfilled** and all implications must be understood before you start changing the hierarchy of your sites. Any deviation from the recommended steps must be carefully considered and is best done in cooperation with a CoreMedia consultant.

Due to the complexity of this topic, the description will be based on an example hierarchy of sites as sketched in [Section 4.10.1, “Example Hierarchy” \[58\]](#). It is your task to adapt the steps to your specific site hierarchy and amount of content items to be moved.

Disclaimer

All the following steps are based on the example hierarchy and, for example, do not incorporate possible required adjustments in external systems. So, ensure to align the suggested steps with your specific requirements.



Carefully Design Multi-Site Hierarchy

As already mentioned in [Section 3.1, “Designing Site Hierarchies” \[16\]](#), changing the hierarchy of your sites is a challenging task. Make sure you have a clear understanding of the implications of your changes before you start.



4.10.1 Example Hierarchy

For the following description the original and target site hierarchy are assumed to be as shown in [Table 4.2, “Site Hierarchy Transformation” \[58\]](#).

Before	After
• en-IE — English (Ireland) • en-GB — English (United Kingdom) • en-FR — English (France) • fr-FR — French (France) • en-DE — English (Germany) • de-DE — German (Germany)	• en-GB • en-IE • en-FR • fr-FR • en-DE • de-DE

Table 4.2. Site Hierarchy Transformation

In short, it is *just* about switching roles between the root-master site `en-IE` (English (Ireland)) and one of its directly derived sites `en-GB` (English (United Kingdom)) below it.

The site type is irrelevant for the actual process, but of course a decision needs to be made in the course of the transformation, like especially, what site type the old root-master site `en-IE` will have after the transformation.

4.10.2 Moving Sites – General Concept

Compared to moving content items, moving sites is a very different story. In short, you do not move any content items, but just change the master-links of the content items within the site.

Special Role – The Site Indicator: Even more, strictly speaking, moving a site in the hierarchy is just about changing the master-links of the site indicator documents. Just doing this, though, would break with the multi-site concept and cause subsequent issues in normal editorial work. That is why in the end, all master-links in a site must be aligned with the site-indicator, thus, represent the same master-derived relationship.

Still, having this distinction between the site indicator, and the other content items is important when it comes to the order of when to apply master-link

changes. The general advice is that **it is always the site indicator, which is the last document in a given site where a master-link is adapted.**

Challenge: Missing Links — Gaps in Content's Master Links

It is perfectly valid, if some documents do not have a master link in derived sites, which is, when there is no counterpart in the master site.

The challenge is best described by example: Assume that you start your transformation process with two documents "Easter Campaign" in the sibling sites `en-GB` and `en-FR`, that are not represented in the current root-master site `en-IE`.

Challenge I: The API, for example in `SitesService` that you will use during the transformation process will not be able to identify these as variants of each other. Thus, you need some *good guess* (like by "same name" assumption) to identify these.

Challenge II: For frictionless editorial work after the transformation, you will want to add a master-derived relationship between these two documents at the end of the transformation process.

A possible option to take is described in next section, where we *repair* the gap of a specific document before the transformation starts: [en-IE: Create Translation Settings \[60\]](#).

4.10.3 Apply New Hierarchy

Fulfill Preconditions

Before you apply the hierarchy change, make sure you have fulfilled all preconditions mentioned in [Section 4.8, "Preconditions for Comprehensive Changes" \[53\]](#).



For the following steps, it is important that you have read [section 4.10.2 \[58\]](#) to learn about the general concept of moving sites within the site hierarchy.

Overview of the steps to apply the new hierarchy along with a short excerpt:

1. [en-IE: Create Translation Settings \[60\]](#)

For the current root-master site `en-IE` create a yet artificial translation settings document and choose your target site type (*translation* or *synchronization*) for the target structure.

2. [en-FR, en-DE: Move Child Sites \[61\]](#)

Multi-Site Challenges | Apply New Hierarchy

Move the directly derived sites `en-FR` and `en-DE` to below the target root-master site `en-GB`.

3. `en-IE`, `en-GB`: Flip Roles [61]

Make the site `en-GB` the root-master site by moving the site `en-IE` below it. It is recommended to prevent site cycles during this process by first removing master-links in `en-GB` prior to setting the new master-links in `en-IE`.

4. All Sites: Review/Set/Remove Site Type [62]

Consider reviewing the site type for all sites in the hierarchy. Eventually, remove the site type indication from the new root-master site `en-GB`.

5. `en-GB`: Cleanup (Optional) [62]

Check and unset irrelevant properties like `ignoreUpdates` and the master-version in the new root-master site.

6. All Sites: Up-to-Date [63]

Align all your master versions to signal translation state *up-to-date*.

7. Validate Your Multi-Site State [63]

Validate that no obstacles exist for your editorial team to pick up work in the new site hierarchy like running the tool `cm validate-multisite`.

8. Publish New Hierarchy [63]

Publish changes near-term and possibly in a staged manner.

And eventually: **Done: Final Remarks** [64].

Must not Use *CoreMedia Studio*

An endeavor like this must not be done in *CoreMedia Studio*. It is not only, that the complexity of this task is too high, to keep track of all required adaptations, but also, that *CoreMedia Studio* makes some assumptions on normal editorial work, that will collide with the required adaptations during this process.

Also, any automated process will help to evaluate the process on test systems before applying it to the production system.



`en-IE`: Create Translation Settings

As described in [Section 4.9.1, "The Site Type" \[55\]](#) the site type is stored in a settings document at `Options/Settings/Translation`. In a typical setup, it is just the root-master site that does not have a translation settings document. Thus, at some point in time you need to decide on the site type for the site `en-IE` now that it becomes a derived site. Doing so now, at the very beginning, will

help restructuring your site, as it helps to find required related variants of these settings to adjust across all sites.

To create this (yet) artificial settings document, follow these steps:

- Create a corresponding settings document for your site `en-IE` and choose your desired target type (translation or synchronization).
- For all directly derived sites of `en-IE`, update their corresponding settings documents to link to that master document. Updating the master version is not required, thus, it may be left empty.

en-FR, en-DE: Move Child Sites

Move the child sites `en-FR` and `en-DE` to below the site `en-GB`. This will make the site `en-GB` the new master site. The sites can be moved one by one.

You can use the Sites Service and the corresponding `ContentSiteAspect` to determine, which contents between the sites are related to find the new master link to set.

As a result, the site hierarchy will look like this:

- `en-IE`
 - `en-GB`
 - `en-FR`
 - `fr-FR`
 - `en-DE`
 - `de-DE`

Handle Nested Derived Sites Later: You will notice, that the nested derived sites (like `fr-FR`) will now be marked as outdated. This is due to the fact, that their respective master content items got updated regarding their properties for master and master version. As more changes will be applied next, there is no need to repair this state now. It will be done later.

en-IE, en-GB: Flip Roles

In this step, we have to touch two sites in parallel, thus, `en-GB` and `en-IE`. While the general process is similar as described in [section "en-FR, en-DE: Move Child Sites" \[61\]](#) and the general pattern to apply in [Section 4.10.2, "Moving Sites – General Concept" \[58\]](#), there is one highly recommended difference to make, which is to prevent intermediate cycles especially in the site hierarchy.

To do so, the recommended order of steps is:

1. **Remember relation between en-GB and en-IE content items:** Remember the master-link to later set at the en-IE before literally *destroying* the relation as intermediate next step.
2. **Remove master-link in en-GB content item:** Meant to prevent an accidental cyclic master-derived relationship.
3. **Set master-link to related en-GB content item:** Update the master-link to the previously remembered content item in the first step.

As described in [Section 4.10.2, "Moving Sites – General Concept" \[58\]](#) it is recommended to process the site indicator document as the last document as you are able to process all other documents in the site one by one, still being able to find their related variants. This is especially important for larger sites in terms of documents contained.

As a result, the site hierarchy will look like this:

- en-GB
 - en-IE
 - en-FR
 - fr-FR
 - en-DE
 - de-DE

All Sites: Review/Set/Remove Site Type

If you followed [section "en-IE: Create Translation Settings" \[60\]](#) the site en-IE is now a derived site and already has the desired site type set.

It is now a good time to review the site types of en-FR and en-DE. In the new structure, you may want to decide differently, like while they may have been *translation* sites before, they now may be better off as *synchronization* sites. For details, read [Section 4.9, "Changing the Site Type" \[55\]](#). The preconditions should already be fulfilled (like all workflows stopped and content items were up-to-date before the change), but you may want to check them again.

Remove Site Type for en-GB: As en-GB is now a root-master site, the site type setting is not needed and should be removed. Ensure, to remove master links towards that document before doing so.

en-GB: Cleanup (Optional)

If not done before clean up the site en-GB by removing the master-versions of the content items as well as unsetting any possibly set `ignoreUpdates` flag. The latter must be done via the Unified API, as the setting is not visible in the *CoreMedia Studio* user interface for root-master sites.

Multi-Site Challenges | Apply New Hierarchy

This step is purely optional, as the master-versions and states of `ignoreUp` dates are not relevant for root-master sites. Still, they may cause issues in subsequent editorial work, like when such contents get copied to other sites (rather than using the recommended workflow based propagation across sites).

All Sites: Up-to-Date

Optionally Prevent Future Conflicts

When changing from *translation* to *synchronization*, consider running a *Synchronization* workflow with *Overwrite Target Site* as conflict resolution parameter to make sure that diverged properties in the derived site are in sync.



Now you benefit from the precondition, that all content items were marked as up-to-date. You just restore this all up-to-date state by updating the master versions of the content items in the derived sites accordingly.

Note the order of doing so: Starting from root-master site `en-GB` (where no master versions need to be updated), first visit all directly derived sites, and then all nested derived sites as each update to the master version also creates a new version of the document.

Validate Your Multi-Site State

As the previous steps may have introduced issues, it is recommended to validate the multi-site state. This can be done with the command line tool **cm validate-multisite** as described in [Section 3.13.1.1](#), “*Validate Multi-Site*” in *Content Server Manual*.

An excerpt of possible new issues after restructuring are (you will find a description along with the command):

- `MS-VALIDATION-4021` - Missing Translation Settings Property Value
- `MS-VALIDATION-6000` - Content not Used in Derived Site
- `MS-VALIDATION-6001` - Content not Used in Derived Site but some Content Exists
- `MS-VALIDATION-6002` - Content without Master
- `MS-VALIDATION-6003` - Content without Master but some Content Exists
- `MS-VALIDATION-6004` - Root Content with Master

Find possible countermeasures in the description of the command as well as in this manual.

Publish New Hierarchy

While restructuring your site hierarchy most likely is dedicated to changed editorial requirements, you may want to consider publishing the changes to the live

system now or near-term. Due to the most likely high amount of contents changed, you may want to consider a staged rollout.

How to stage the rollout should be planned as carefully as the internal changes to the site hierarchy. Assuming that it is ok to risk temporarily unavailable possibilities to switch between language variants in your deployed site, this publication order (per site) may be one possible option:

1. `en-GB`
2. `en-IE`
3. all other sites

This prevents the risk of introducing a cyclic master-derived relationship, which may happen if the site `en-IE` is published first. Just as described in [Section 4.10.2, “Moving Sites – General Concept” \[58\]](#) it may be beneficial to process the site indicator document as the last document to be published for each site.

Thus, it makes sense, to carefully think about the publication order in similar small steps and intermediate states as sketched here.

Done: Final Remarks

During the transformation you will create mildly irregular states (such as the link to the root-document not being updated for the temporary site). These irregular states are now gone, so that your editors may start working again with the new site hierarchy. Your editors should be aware though, that updates to the translation state in *CoreMedia Studio* will take some time to be fully propagated, which means, for example, that library filters may not provide the expected results until these updates have been propagated.

Glossary

Blob	Binary Large Object or short blob, a property type for binary objects, such as graphics.
CaaS	Content as a Service or short caas, a synonym for the CoreMedia Headless Server.
CAE Feeder	Content applications often require search functionality not only for single content items but for content beans. The <i>CAE Feeder</i> makes content beans searchable by sending their data to the <i>Search Engine</i> , which adds it to the index.
Content Application Engine (CAE)	The <i>Content Application Engine</i> (CAE) is a framework for developing content applications with <i>CoreMedia CMS</i>. While it focuses on web applications, the core frameworks remain usable in other environments such as standalone clients, portal containers or web service implementations. The CAE uses the Spring Framework for application setup and web request processing.
Content Bean	A content bean defines a business oriented access layer to the content, that is managed in <i>CoreMedia CMS</i> and third-party systems. Technically, a content bean is a Java object that encapsulates access to any content, either to CoreMedia CMS content items or to any other kind of third-party systems. Various CoreMedia components like the CAE Feeder or the data view cache are built on this layer. For these components the content beans act as a facade that hides the underlying technology.
Content Delivery Environment	The <i>Content Delivery Environment</i> is the environment in which the content is delivered to the end-user. It may contain any of the following modules: • <i>CoreMedia Master Live Server</i> • <i>CoreMedia Replication Live Server</i> • <i>CoreMedia Content Application Engine</i> • <i>CoreMedia Search Engine</i> • <i>Elastic Social</i>

- *CoreMedia Native Personalization*

Content Feeder	The <i>Content Feeder</i> is a separate web application that feeds content items of the CoreMedia repository into the <i>CoreMedia Search Engine</i> . Editors can use the <i>Search Engine</i> to make a full text search for these fed items.
Content item	In <i>CoreMedia CMS</i> , content is stored as self-defined content items. Content items are specified by their properties or fields. Typical content properties are, for example, title, author, image and text content.
Content Management Environment	The <i>Content Management Environment</i> is the environment for editors. The content is not visible to the end user. It may consist of the following modules: • <i>CoreMedia Content Management Server</i> • <i>CoreMedia Workflow Server</i> • <i>CoreMedia Studio</i> • <i>CoreMedia Search Engine</i> • <i>CoreMedia Native Personalization</i> • <i>CoreMedia Preview CAE</i>
Content Management Server	Server on which the content is edited. Edited content is published to the Master Live Server.
Content Repository	<i>CoreMedia CMS</i> manages content in the Content Repository. Using the Content Server or the UAPI you can access this content. Physically, the content is stored in a relational database.
Content Server	<i>Content Server</i> is the umbrella term for all servers that directly access the CoreMedia repository: <i>Content Servers</i> are web applications running in a servlet container. • <i>Content Management Server</i> • <i>Master Live Server</i> • <i>Replication Live Server</i>
Content type	A content type describes the properties of a certain type of content. Such properties are for example title, text content, author, ...
Contributions	Contributions are tools or extensions that can be used to improve the work with <i>CoreMedia CMS</i> . They are written by CoreMedia developers – be it clients, partners or Core-

	Media employees. CoreMedia contributions are hosted on Github at https://github.com/coremedia-contributions .
Control Room	<i>Control Room</i> is a <i>Studio</i> plugin, which enables users to manage projects, work with workflows, and collaborate by sharing content with other <i>Studio</i> users.
CORBA (Common Object Request Broker Architecture)	The term <i>CORBA</i> refers to a language- and platform-independent distributed object standard which enables interoperation between heterogenous applications over a network. It was created and is currently controlled by the Object Management Group (OMG), a standards consortium for distributed object-oriented systems. CORBA programs communicate using the standard IIOP protocol.
CoreMedia Studio	<i>CoreMedia Studio</i> is the working environment for business specialists. Its functionality covers all the stages in a web-based editing process, from content creation and management to preview, test and publication. As a modern web application, <i>CoreMedia Studio</i> is based on the latest standards like Ajax and is therefore as easy to use as a normal desktop application.
Dead Link	A link, whose target does not exist.
Derived Site	A derived site is a site, which receives localizations from its master site. A derived site might itself take the role of a master site for other derived sites.
DTD	A Document Type Definition is a formal context-free grammar for describing the structure of XML entities. The particular DTD of a given Entity can be deduced by looking at the document prolog: <pre><!DOCTYPE coremedia SYSTEM "http://www.coremedia.com/dtd/coremedia.dtd"</pre> There're two ways to indicate the DTD: Either by Public or by System Identifier. The System Identifier is just that: a URL to the DTD. The Public Identifier is an SGML Legacy Concept.
Elastic Social	<i>CoreMedia Elastic Social</i> is a component of <i>CoreMedia CMS</i> that lets users engage with your website. It supports features like comments, rating, likings on your website. <i>Elastic Social</i> is integrated into <i>CoreMedia Studio</i> so editors can moderate user generated content from their common

workplace. *Elastic Social* bases on NoSQL technology and offers nearly unlimited scalability.

EXML

EXML is an XML dialect used in former CoreMedia Studio version for the declarative development of complex Ext JS components. EXML is Jangaroo 2's equivalent to Apache Flex (formerly Adobe Flex) MXML and compiles down to ActionScript. Starting with release 1701 / Jangaroo 4, standard MXML syntax is used instead of EXML.

Folder

A folder is a resource in the CoreMedia system which can contain other resources. Conceptually, a folder corresponds to a directory in a file system.

FTL

FTL (FreeMarker Template Language) is a Java-based template technology for generating dynamic HTML pages.

gRPC

gRPC is an open source high performance Remote Procedure Call (RPC) framework.

Headless Server

CoreMedia Headless Server is a CoreMedia component introduced with CoreMedia Content Cloud which allows access to CoreMedia content as JSON through a GraphQL endpoint.

The generic API allows customers to use CoreMedia CMS for headless use cases, for example delivery of pure content to Native Mobile Applications, Smartwatches/Wearable Devices, Out-of-Home or In-Store Displays or Internet-of-Things use cases.

Home Page

The main entry point for all visitors of a site. Technically it is often referred to as root document and also serves as provider of the default layout for all subpages.

IETF BCP 47

Document series of *Best current practice* (BCP) defined by the Internet Engineering Task Force (IETF). It includes the definition of IETF language tags, which are an abbreviated language code such as en for English, pt-BR for Brazilian Portuguese, or nan-Hant-TW for Min Nan Chinese as spoken in Taiwan using traditional Han characters.

IOR (Interoperable Object Reference)

A CORBA term, *Interoperable Object Reference* refers to the name with which a CORBA object can be referenced.

Jangaroo

Jangaroo is a JavaScript framework developed by CoreMedia that supports TypeScript (formerly MXML/ActionScript) as an input language which is compiled down to JavaScript compatible with Ext JS. You will find detailed descriptions on the Jangaroo webpage <http://www.jangaroo.net>. Jangaroo 4 is the Action-

	Script/MXML/Maven based version for CMCC 10. Since CMCC 11 (2110), Jangaroo uses TypeScript and is implemented as a <i>Node.js</i> and <i>npm</i> based set of tools.
Java Management Extensions (JMX)	The Java Management Extensions is an API for managing and monitoring applications and services in a Java environment. It is a standard, developed through the Java Community Process as JSR-3. Parts of the specification are already integrated with Java 5. JMX provides a tiered architecture with the instrumentation level, the agent level and the manager level. On the instrumentation level, MBeans are used as managed resources.
Locale	Locale is a combination of country and language. Thus, it refers to translation as well as to localization. Locales used in translation processes are typically represented as IETF BCP 47 language tags.
Master Live Server	The <i>Master Live Server</i> is the heart of the <i>Content Delivery Environment</i> . It receives the published content from the <i>Content Management Server</i> and makes it available to the CAE. If you are using the <i>CoreMedia Multi-Master Management Extension</i> you may use multiple <i>Master Live Server</i> in a CoreMedia system.
Master Site	A master site is a site other localized sites are derived from. A localized site might itself take the role of a master site for other derived sites.
MIME	With Multipurpose Internet Mail Extensions (MIME), the format of multi-part, multimedia emails and of web documents is standardised.
MXML	MXML is an XML dialect used by Apache Flex (formerly Adobe Flex) for the declarative specification of UI components and other objects. Up to CMCC 10 (2107), CoreMedia Studio used the Open Source compiler Jangaroo 4 to translate MXML and ActionScript sources to JavaScript that is compatible with Ext JS 7. Starting with CMCC 11 (2110), a new, Node.js and npm based version of Jangaroo is used that supports standard TypeScript syntax instead of MXML/ActionScript, still compiling to Ext JS 7 JavaScript.
OCI (Open Container Initiative)	The Open Container Initiative (OCI) is a Linux Foundation project that defines open industry standards for container formats and runtimes. OCI specifications ensure compatibility and interoperability between container tools, engines, and orchestration platforms like Docker and Kubernetes.

ORAS (OCI Registry As Storage)	ORAS (OCI Registry As Storage) is a tool and specification that extends OCI registries to store and distribute OCI artifacts beyond container images. It provides a standardized way for developers to push and pull arbitrary content types to and from container registries, enabling these registries to function as general artifact stores.
Personalisation	On personalised websites, individual users have the possibility of making settings and adjustments which are saved for later visits.
Projects	With projects you can group content and manage and edit it collaboratively, setting due dates and defining to-dos. Projects are created in the Control Room and managed in project tabs.
Property	In relation to CoreMedia, properties have two different meanings: In CoreMedia, content items are described with properties (content fields). There are various types of properties, e.g. strings (such as for the author), Blobs (e.g. for images) and XML for the textual content. Which properties exist for a content item depends on the content type. In connection with the configuration of CoreMedia components, the system behavior of a component is determined by properties.
Replication Live Server	The aim of the <i>Replication Live Server</i> is to distribute load on different servers and to improve the robustness of the <i>Content Delivery Environment</i> . The <i>Replication Live Server</i> is a complete Content Server installation. Its content is an replicated image of the content of a <i>Master Live Server</i> . The <i>Replication Live Server</i> updates its database due to change events from the <i>Master Live Server</i> . You can connect an arbitrary number of <i>Replication Live Servers</i> to the <i>Master Live Server</i> .
Resource	A folder or a content item in the CoreMedia system.
ResourceURI	A <i>ResourceUri</i> uniquely identifies a page which has been or will be created by the <i>Active Delivery Server</i> . The <i>ResourceUri</i> consists of five components: Resource ID, Template ID, Version number, Property names and a number of key/value pairs as additional parameters.
Responsive Design	Responsive design is an approach to design a website that provides an optimal viewing experience on different devices, such as PC, tablet, mobile phone.

Site	A site is a cohesive collection of web pages in a single locale, sometimes referred to as localized site. In <i>CoreMedia CMS</i> a site especially consists of a site folder, a site indicator and a home page for a site. A typical site also has a master site it is derived from.
Site Folder	All contents of a site are bundled in one dedicated folder. The most prominent document in a site folder is the site indicator, which describes details of a site.
Site Indicator	A site indicator is the central configuration object for a site. It is an instance of a special content type, most likely <code>CMSite</code>.
Site Manager Group	Members of a site manager group are typically responsible for one localized site. Responsible means that they take care of the contents of that site and that they accept translation tasks for that site.
Template	In CoreMedia, FreeMarker templates used for displaying content are known as Templates. OR In <i>Blueprint</i> a template is a predeveloped content structure for pages. Defined by typically an administrative user a content editor can use this template to quickly create a complete new page including, for example, navigation, predefined layout and even predefined content.
Translation Manager Role	Editors in the translation manager role are in charge of triggering translation workflows for sites.
User Changes Application	The <i>User Changes Application</i> is a <i>Content Repository</i> listener, which collects all content, modified by <i>Studio</i> users. This content can then be managed in the <i>Control Room</i>, as a part of projects and workflows.
Variants	The set of all content items in a multi-site hierarchy related to each other via master references. This includes the top-level master content items themselves.
Version history	A newly created content item receives the version number 1. New versions are created when the content item is checked in; these are numbered in chronological order.
Weak Links	In general <i>CoreMedia CMS</i> always guarantees link consistency. But links can be declared with the <i>weak</i> attribute, so that they are not checked during publication or withdrawal.

Caution! Weak links may cause dead links in the live environment.

Workflow

A workflow is the defined series of tasks within an organization to produce a final outcome. Sophisticated applications allow you to define different workflows for different types of jobs. So, for example, in a publishing setting, a document might be automatically routed from writer to editor to proofreader to production. At each stage in the workflow, one individual or group is responsible for a specific task. Once the task is complete, the workflow software ensures that the individuals responsible for the next task are notified and receive the data they need to execute their stage of the process.

Workflow Server

The *CoreMedia Workflow Server* is part of the Content Management Environment. It comes with predefined workflows for publication but also executes freely definable workflows.

XLIFF

XLIFF is an XML-based format, standardized by OASIS for the exchange of localizable data. An XLIFF file contains not only the text to be translated but also metadata about the text. For example, the source and target language. *CoreMedia Studio* allows you to export content items in the XLIFF format and to import the files again after translation.

Index

C

CLDR, 27
CleanInTranslation, 33

D

derived, 13
design, **15**
 locales, **22**

I

IETF BCP 47, 22

L

language tag, 22
locale, 22
 CLDR, 27
 IETF BCP 47, 22
 language tag, 22
localization
 synchronization, 16, 55
 translation, 16, 55

M

master, 13

N

nagbar, 30
navigation
 translation, 43

R

referrer relations, 43

S

site, 14
 type, 55
 change, 55
 synchronization, 16, 55
 translation, 16, 55
site type, 55
 change, 55
 synchronization, 55
 translation, 55
sites
 hierarchy, **16**, 57
 abstract site, 17
 country first, 17
 language first, 17
 recommended, 20
synchronization, 16, 55

T

taxonomy
 translation, 43
translation, 16, 55
 referrer relations, 43
translationStrategies
 LANG_TRANSLATION, 55
 SYNC, 55

W

workflow, 14