CoreMedia Content Cloud

Analytics Connectors Manual

CoreMedia

CoreMedia GmbH

Altes Klöpperhaus, 5. OG

Rödingsmarkt 9

20459 Hamburg

April 25, 2023 (Release 2301)

List of Figures

List of Tables

1. Preface

This manual describes the *CoreMedia Analytics Connectors*.

A general overview of analytics and how it is integrated into *CoreMedia Blueprint* is given in Chapter 2, *Overview* [14]. Tracking is described in Chapter 3, *Tracking* [15]. The description elaborates on the aspects configuration, assembling of tracking data and firing of tracking calls. Chapter 4, *Retrieval* [22] is dedicated to the topic of "top-n-lists", for example content lists that correspond to the top n entries of an analytics report. This manual closes with notes on how to integrate another third-party analytics solution in Chapter 5, *Integrating an Analytics Service Provider* [31].

1.1 Audience

This manual is intended for all readers who are interested to use one of the integrated third-party analytics solutions or to integrate a new one with *CoreMedia CMS*.

1.2 Typographic Conventions

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

Element	Typographic format	Example
Source code Command line entries Parameter and values	Monospaced Font	`cm contentserver start`
Menu names and entries	Bold, linked with |	Open the menu entry **Format|Normal**
Field names CoreMedia Components	Italic	Enter in the field *Heading* The *CoreMedia Component*
Entries	In quotation marks	Enter "On"
(Simultaneously) pressed keys	Bracketed in "<>", linked with "+"	Press the keys <Ctrl>+<A>
Emphasis	Italic	It is *not* saved
Buttons	Bold, with square brackets	Click on the **[OK]** button
Code lines in code examples which continue in the next line	\	`cm contentserver \` `start`

Table 1.1. Typographic conventions

In addition, these symbols can mark single paragraphs:

Pictograph	Description
	Tip: This denotes best practices or recommendations.

Pictograph	Description
	Warning: Please pay special attention to the text.
	Danger: The violation of these rules causes severe damage.

Table 1.2. Pictographs

1.3 CoreMedia Services

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

NOTE

CoreMedia User Orientation for CoreMedia Developers and Partners

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

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

- Section 1.3.1, “Registration” [5] describes how to register for the usage of the services.
- Section 1.3.2, “CoreMedia Releases” [6] describes where to find the download of the software.
- Section 1.3.3, “Documentation” [7] describes the CoreMedia documentation. This includes an overview of the manuals and the URL where to find the documentation.
- Section 1.3.4, “CoreMedia Training” [10] describes CoreMedia training. This includes the training calendar,the curriculum and certification information.
- Section 1.3.5, “CoreMedia Support” [10] describes the CoreMedia support.

1.3.1 Registration

In order to use CoreMedia services you need to register. Please, start your initial registration via the CoreMedia website. Afterwards, contact the CoreMedia Support (see Section 1.3.5, “CoreMedia Support” [10]) by email to request further access depending on your customer, partner or freelancer status so that you can use the CoreMedia services.

1.3.2 CoreMedia Releases

Downloading and Upgrading the Blueprint Workspace

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

https://releases.coremedia.com/cmcc-11

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

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

Maven artifacts

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

https://repository.coremedia.com

You have to add your CoreMedia credentials to your Maven settings file as described in section Section 3.1, "Prerequisites" in *Blueprint Developer Manual* .

npm packages

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

https://npm.coremedia.io

Your pnpm client first needs to be logged in to be able to utilize the registry (see Section 3.1, "Prerequisites" in *Blueprint Developer Manual*).

License files

You need license files to run the CoreMedia system. Contact the support (see Section 1.3.5, "CoreMedia Support" [10]) to get your licences.

1.3.3 Documentation

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

https://documentation.coremedia.com

The manuals have the following content and use cases:

Manual	Audience	Content
Adaptive Personalization Manual	Developers, architects, administrators	This manual describes the configuration of and development with *Adaptive Personalization*, the CoreMedia module for personalized websites. You will learn how to configure the GUI used in *CoreMedia Studio*, how to use predefined contexts and how to develop your own extensions.
Analytics Connectors Manual	Developers, architects, administrators	This manual describes how you can connect your CoreMedia website with external analytic services, such as Google Analytics.
Blueprint Developer Manual	Developers, architects, administrators	This manual gives an overview over the structure and features of *CoreMedia Content Cloud*. It describes the content type model, the *Studio* extensions, folder and user rights concept and many more details. It also describes administrative tasks for the features. It also describes the concepts and usage of the project workspace in which you develop your CoreMedia extensions. You will find a description of the Maven structure, the virtualization concept, learn how to perform a release and many more.
Connector Manuals	Developers, administrators	This manuals gives an overview over the use cases of the eCommerce integration. It describes the deployment of the Commerce Connector and how to connect it with the CoreMedia and eCommerce system.
Content Application Developer Manual	Developers, architects	This manual describes concepts and development of the *Content Application Engine (CAE)*. You will learn how to write JSP or Freemarker templates that access the other CoreMedia modules and use the sophisticated caching mechanisms of the CAE.

Manual	Audience	Content
Content Server Manual	Developers, architects, administrators	This manual describes the concepts and administration of the main CoreMedia component, the *Content Server*. You will learn about the content type model which lies at the heart of a CoreMedia system, about user and rights management, database configuration, and more.
Deployment Manual	Developers, architects, administrators	This manual describes the concepts and usage of the CoreMedia deployment artifacts. That is the deployment archive and the Docker setup. You will also find an overview of the properties required to configure the deployed system.
Elastic Social Manual	Developers, architects, administrators	This manual describes the concepts and administration of the *Elastic Social* module and how you can integrate it into your websites.
Frontend Developer Manual	Frontend Developers	This manual describes the concepts and usage of the Frontend Workspace. You will learn about the structure of this workspace, the CoreMedia themes and bricks concept, the CoreMedia Freemarker facade API, how to develop your own themes and how to upload your themes to the CoreMedia system.
Headless Server Developer Manual	Frontend Developers, administrators	This manual describes the concepts and usage of the *Headless Server*. You will learn how to deploy the Headless Server and how to use its endpoints for your sites.
Importer Manual	Developers, architects	This manual describes the structure of the internal CoreMedia XML format used for storing data, how you set up an *Importer* application and how you define the transformations that convert your content into CoreMedia content.
Multi-Site Manual	Developers, Multi-Site Administrators, Editors	This manual describes different otions to desgin your site hierarchy with several languages. It also gives guidance to avoid common pitfalls during your work with the multi-site feature.

Manual	Audience	Content
Operations Basics Manual	Developers, administrators	This manual describes some overall concepts such as the communication between the components, how to set up secure connections, how to start application.
Search Manual	Developers, architects, administrators	This manual describes the configuration and customization of the *CoreMedia Search Engine* and the two feeder applications: the *Content Feeder* and the *CAE Feeder*.
Site Manager Developer Manual	Developers, architects, administrators	This manual describes the configuration and customization of *Site Manager*, the Java based stand-alone application for administrative tasks. You will learn how to configure the *Site Manager* with property files and XML files and how to develop your own extensions using the *Site Manager API*. The Site Manager is deprecated for editorial work.
Studio Developer Manual	Developers, architects	This manual describes the concepts and extension of *CoreMedia Studio*. You will learn about the underlying concepts, how to use the development environment and how to customize *Studio* to your needs.
Studio User Manual	Editors	This manual describes the usage of *CoreMedia Studio* for editorial and administrative work. It also describes the usage of the *Adaptive Personalization* and *Elastic Social* GUI that are integrated into *Studio*.
Studio Benutzerhandbuch	Editors	The Studio User Manual but in German.
Supported Environments	Developers, architects, administrators	This document lists the third-party environments with which you can use the CoreMedia system, Java versions or operation systems for example.
Unified API Developer Manual	Developers, architects	This manual describes the concepts and usage of the *CoreMedia Unified API*, which is the recommended API for most applications. This includes access to the content repository, the workflow repository and the user repository.

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

Table 1.3. CoreMedia manuals

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

Email: documentation@coremedia.com

1.3.4 CoreMedia Training

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

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

http://www.coremedia.com/training

Contact the training department at the following email address:

Email: training@coremedia.com

1.3.5 CoreMedia Support

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

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

http://support.coremedia.com/

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

Email: support@coremedia.com

Create a support request

Support request

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

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

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

Support checklist

To put it in a nutshell, CoreMedia needs:

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

Log files

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

Which Log File?

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

Where do I Find the Log Files?

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

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

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

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

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

1.4 Changelog

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

Section	Version	Description

Table 1.4. Changes

2. Overview

CoreMedia Analytics Connectors demonstrates how to integrate third-party analytics services into *CoreMedia CMS*. The following third-party analytics services are integrated:

- Google Analytics

The integration extends the delivery side (*CAE*) with tracking of page impressions. To take advantage of tracked data, this contribution extends the content types to support "top-n-lists" based on tracking data. A "top-n-list" content aggregates a list of "n" top-performing contents - regarding page impressions - where "n" is the number of resulting content items to be displayed. *CoreMedia Studio* is extended with content forms to set up top-n-lists and to configure the external analytics services.

If your project is based on *CoreMedia Blueprint*, you will be able to use the integration out of the box. Otherwise, it serves as an example of how to integrate external analytics services into your CoreMedia project.

CoreMedia Analytics Connectors combines the following major components:

- *CoreMedia Content Server* content types to define "top-n-lists"
- *CAE* content beans and service beans to render "top-n-lists"

  Services include retrieval of analytics data, generation of "top-n-lists" and access to analytics related settings.
- Solr external file field exporters
- a plugin for *CoreMedia Studio*

  The plugin allows editors to edit "top-n-list" content items and to configure the analytics service specific parameters to enable tracking and retrieval of tracked data.

3. Tracking

Tracking user actions on a website is typically implemented by adding calls to vendor specific JavaScript functions to the pages of the site. These functions populate a data structure which is eventually sent to the analytics service via a HTTP request for an invisible image, also known as tracking pixel.

The data being sent to the analytics service includes data about the following topics:

- Content being displayed on the page

In addition, each analytics service requires some way of identifying the account the incoming data is to be written to.

The tracking configuration is stored in the content repository within the `settings` property of navigation content items. Note that settings can be linked to a content and content objects inherit settings defined for their navigation contexts (channels). So settings defined at a navigation override settings defined at its parent navigation. For a Page content bean named page, the Google Analytics configuration, for example, is stored under the property path `page.settings.googleAnalytics`.

The configuration options for tracking are described further down in Section 3.4, “Studio Integration” [19].

3.1 Assembling Tracking Information

CoreMedia Analytics Connectors provides a `ViewHookEventListener` and an `AnalyticsProvider` implementation for each integrated third-party service provider. The `ViewHookEventListener` reacts on a `ViewHookEvent` of type `head` for content of type `Page`. If sufficient configuration is available, it renders the corresponding provider's JavasSript into the `head` section of the `Page`. The provider specific `AnalyticsProvider` implementation provides access to the basic configuration that is necessary to establish a tracking connection to that particular service. Most importantly, the `AnalyticsProvider` implementation can check if any required properties are missing and suppresses rendering of any output for that service if it is not properly configured.

Note that the integration of analytics extensions is controlled by the CoreMedia Extension Tool. Per default, all analytics extensions are active but can be turned off by using this tool (see Section 4.1.5, "Project Extensions" in *Blueprint Developer Manual*).

See the Javadoc for more details on which properties the analytics listeners and interceptors provide for tracking and Section 3.4, "Studio Integration" [19] for details on how to configure them.

Consult the Content Application Developer Manual for information on how to register interceptors and `ViewHookEventListener`s.

3.2 Views

As explained in the previous section, *CAE* `ViewHookEventListener`s are used to make data to be tracked (and the tracking configuration itself) accessible when rendering a view. This data is used in views to build the tracking calls. *CAE* `ViewHookEventListener`s react on rendering of `com.coremedia.blueprint.common.contentbeans.Page` beans to add code into the head of a page. The analytics integration uses the `head` view that serves the following purposes:

- Includes third-party tracking libraries
- Sets up vendor-specific JavaScript data structures required for tracking
- Includes the asynchronous calls to the JavaScript tracking code

`GoogleAnalytics.head.ftl` includes JavaScript libraries specific for an external service and converts the tracking data into JavaScript objects used by the final tracking calls. The FTL checks if the service provider that it implements is enabled for the current page, that is, if it is properly configured and not explicitly disabled.

Page view tracking calls use the property "`enabled` " to check whether the third-party service is enabled. Setting this property explicitly to "false", disables the service provider, while setting it to "true" will only enable further processing of the provider's configuration (which might ultimately enable that service if the configuration is complete).

3.3 JavaScript Code

Tracking is performed by calling vendor-specific JavaScript functions. *CoreMedia Analytics Connectors* offers a thin layer around these.

Page view tracking calls are fired when a page is loaded. Corresponding calls are either included by the vendor specific JavaScript files to include, or have to be included in the implementation's `asHead` FTL of its analytics implementation.

The following subsections describe the JavaScript variables that are set in the `head` views of the FTLs described in the previous section.

3.3.1 Google Analytics

The Google Analytics integration provides an abstraction layer implemented in a JavaScript file `alx-integration-googleanalytics.js` which contains constructors for objects holding common data and functions to track page views and events:

- `GaAccountData` for the web property id and domain names,
- `GaPageviewData` for data related to a page view, such as the URL and the content id, and
- `GaEventData` for data related to an observed event, such as event category and label.

These data objects are supplied to the implemented functions

- `gaTrackPageview` for tracking page views, and
- `gaTrackEvent` for tracking events.

See JavaScript inline documentation for details.

3.4 Studio Integration

CoreMedia Analytics Connectors can be configured per site and per page. The settings for this can be configured using the struct editor for the property field `localSettings`. Each tracking provider is configured in a separate `StructProperty`. An example of the tracking configuration is shown below.

```
<StringProperty Name="analyticsProvider">googleAnalytics</StringProperty>
<StructProperty Name="googleAnalytics">
  <Struct>
    <StringProperty Name="webPropertyId">UA-XXXXXXX-1</StringProperty>
    <StringProperty Name="domainName">auto</StringProperty>
  </Struct>
</StructProperty>
```

> **NOTE**
>
> Note that tracking can be temporarily disabled for any service provider (even for a particular page) by adding a Boolean property `disabled` with value `true` to the provider's struct property.

The `AnalyticsSettingsProvider` implementations can expose settings for Studio components. These can, for example, be used to provide deep links for service provider reports. See Javadoc for details.

> **NOTE**
>
> In the following section describes the property names to be used in the generic struct editor of the *CoreMedia Studio*.

3.4.1 Google Analytics

The integration of Google Analytics not only allows you to configure tracking but also to configure the preview toolbar button to open a Google Analytics report drilled down to the active content object. These aspects are covered by the options presented in the following tables.

Technical Variable Name	Description/Value	Required
`webPropertyId`	The Google Analytics web property ID to track to. The Web Property ID has the format `UA-12345678-1`	true

Technical Variable Name	Description/Value	Required
`domainName`	The domain name to be sent to Google Analytics, if you are running a multi-site environment.	false
`disableAdvertisingFeaturesPlugin`	Disables the Advertising Features Plugin. If you do not need Google Analytics' advertising features, the plugin should be disabled. The plugin might create additional cookies if it is not disabled. See https://developers.google.com/analytics/devguides/collection/analyticsjs/display-features for more information.	false

Table 3.1. Google Analytics Tracking Configuration Options

Technical Variable Name	Description/Value	Required
`pageReport`	The name of the report to create the drill down URL for. Defaults to `content-pages`	false
`accountId`	Your numeric Google Analytics account ID to create the drill down URL for. See the tip below.	false
`wpid`	Your numeric Google Analytics web property ID to create the drill down URL for. See the paragraph below.	false
`pid`	Your numeric Google Analytics profile ID to create the drill down URL for. See the paragraph below.	false

Table 3.2. Google Analytics Studio Configuration Options

The configuration for Studio integration should be applied in InternalAnalyticsSettings, see Internal Settings [25]. Example:

```
<StructProperty Name="googleAnalytics">
  <Struct>
    <StringProperty Name="pageReport">content-pages</StringProperty>
    <IntProperty Name="accountId">12345678</IntProperty>
    <IntProperty Name="wpid">12345678</IntProperty>
    <IntProperty Name="pid">12345678</IntProperty>
```

```
  </Struct>
</StructProperty>
```

NOTE

Given a Google Analytics drill down URL, you may notice the URL a hash parameter fragment of the form `/aXXXXXXXXwYYYYYYYYpZZZZZZZZ/`. In this case, `XXXXXXXX` is your 8 digit account ID, `YYYYYYYY` is your 8 digit web property ID and `ZZZZZZZZ` is your 8 digit profile ID. Each of these values is required to build a valid report drill down URL.

NOTE

If the *CoreMedia Elastic Social* Extension is enabled and social tracking is configured, `webPropertyId` must be set to the same value as `socialTrackingId` in the *Elastic Social* configuration.

4. Retrieval

Data aggregated by analytics service providers can not only be used to generate tables and diagrams but also to generate "top-n-lists" for use on the delivery side of a content management environment. To generate "top-n-lists" of contents based on their rank within an analytics report, report data is gathered and cached using *CoreMedia Elastic Core* infrastructure.

The following components play key roles in this setting:

- `CMALXBaseList` content objects

  Instances of this content type serve as configuration objects for a retrieval task that fetches the corresponding data. The content beans are also used at rendering time to retrieve the content objects corresponding to the tracking data cached using *CoreMedia Elastic Core*.

- `AnalyticsServiceProvider`

  Implementations actually access the third-party analytics service provider and gather data. Data is persisted using the `CMALXBaseListModelService`. This model service retrieves and stores objects of type `ReportModel` which hold the current configuration of the report, the preprocessed report data and a timestamp to represent the reports freshness.

- `FetchReportsTask`

  An elastic worker task iterates over the "top-n-list" content items of a tenant and uses all `AnalyticsServiceProvider` implementations available in the current Spring context to retrieve data for them. First the task gets all root navigation items and "top-n-list" items for a tenant and executes for "top-n-list" content items and the root content item of the same site. Then task checks, if the corresponding `ReportModel` is too old or differs in its configuration. This ensures that changes in the configuration trigger an almost immediate new retrieval of data. If retrieval is due, data is fetched and passed to the `CMALXBaseList` instance to preprocess the result list. Then the report model is saved.

  There are different types of time intervals involved, which can be confusing:

  - Interval of the `FetchReportsTask` - the task is executed quite often, for example every minute, but only fetches data if necessary.
  - Interval for retrieving data from a specific analytics provider - the effective retrieval interval in which data is actually retrieved if the configuration has not changed,

for example every 180 minutes. It is configured per top-n-list instance using the `interval` property (see Table 4.1, "Generic Retrieval Configuration Options" [23]).

- Time range of the fetched data - usually you only retrieve data for a certain time range, for example you are interested in the report data for the last week.

> **NOTE**
>
> `FetchReportTask` assumes that data is fetched synchronously. If the analytics service provider provides asynchronous access only, you will have to set up additional tasks that fetch the report data. An implementation of `ElasticAnalytcisServiceProvider` should then store information used by the additional tasks (for subsequent calls to the analytics service provider) in the report model and return an empty list themselves.

Top-n-lists (`CMALXBaseList` instances) provide an `analyticsProvider` property that determines the analytics service provider to use.

Retrieval configuration is stored in the settings of the content (or one of its channels). Note that settings defined at a content override settings defined by its channel. For a Page content bean named page, the Google Analytics configuration, for example, is stored under the property path `page.settings.googleAnalytics`.

The following configuration properties apply to all analytics service provider implementations:

Technical Variable Name	Description/Value	Required
`analyticsProvider`	The service key of the analytics provider to use. If not set, your list will be empty, even if data is cached.	false
`maxLength`	The 'n' of a top-n-list (its maximum length). Default is 10.	false
`interval`	The interval to fetch report data. Default is 24 hours.	false
`limit`	The maximum number of records of a fetched report. Default is 1000.	false

Table 4.1. Generic Retrieval Configuration Options

NOTE

Note that retrieval can be temporarily disabled (even for a particular page) by setting the service's `interval` property, for example `googleAnalytics.interval`, to `0`.

Subtypes of `CMALXBaseList` define report types to fetch. The following examples are provided by *CoreMedia Analytics Connectors*:

- `CMALXPageList`

  Instances of this content type refer to a report containing page views. Generic properties used at retrieval time are `documentType` and `baseChannel` limiting the items to display at rendering time. Hence, the "top-n-list" will be made up of content objects of type `documentType` below channel `baseChannel`. The property `defaultContent` defines a list of content objects to be displayed if no report data is available.

- `CMALXEventList`

  Instances of this content type refer to a report containing tracked events. Generic event properties are `category` and `action`. The category is the name you supply for the group of objects to track. The action is a string identifying the type of user interaction to be tracked. The pair of category and action should uniquely identify the event.

On the retrieval side, only implementations of `AnalyticsServiceProvider` are specific to an analytics service provider. In the next sections, the existing service provider implementations are presented.

NOTE

Cached report models are cleared, if the data has not been updated in the last 30 days.

CAUTION

The tables in the following sections use the technical names of configuration options. Look them up in the resource bundles of the corresponding Studio extension modules to find out the localized names of the properties.

CAUTION

The analytics service providers restrict usage in different ways (with respect to request frequency, request count per time unit or response size in terms of record count). Ensure that your configuration matches those limitations.

CAUTION

Some settings can contain secret information and should not be published.

Therefore these internal settings should be separated from settings, that are required on live systems.

Internal analytics settings can be configured for a site in `[site]/Options/Settings/Internal/InternalAnalyticsSettings.xml`.

For the document `InternalAnalyticsSettings` some specific rules apply:

- It should not be published.
- It should not be linked in other documents.
- If a setting is configured in linked/local settings and in InternalAnalyticsSettings, the value of linked/local settings is applied.
- Retrieval tasks: InternalAnalyticsSettings are only applied, if tracking is configured (linked/local settings contain analyticsProvider and provider specific properties).
- Headless Server: The containing folder `[site]/Options/Settings/Internal` should be configured in the `blocklist` to prevent delivery of the folder content in headless environments.

4.1 Google Analytics

Google Analytics is accessed using the `Google Analytics Universal` API. To access Google Analytics Data, you need to create a service account (Google Developer Console). The service account email and p12 Key File (PKCS Standard) are used for OAuth 2.0 authentication.

NOTE

Remember to grant access to the analytics reports for the service account.

Custom variables must be defined as dimension1...n in the Google Analytics Web interface.

Custom Variable	Use
`dimension1`	Content ID
`dimension2`	Content Type
`dimension3`	Navigation Path

Table 4.2. Google Analytics custom variables

Both page views and tracked events are considered at retrieval time. The following configuration options are available:

Technical Variable Name	Description/Value	Required
`applicationName`	Name of the app accessing Google Analytics data, used in the UserAgent header of each request.	true
`pid`	A numeric Google Analytics profile ID providing the data to fetch (compare with property `pid` described in Table 3.1, "Google Analytics Tracking Configuration Options" [19]).	true
`serviceAccountEmail`	Email address of the service account.	true

Technical Variable Name	Description/Value	Required
`p12File`	Create a CMDownload content item and upload the p12 Key File as blob. Link this content item in the Analytics Settings P12 Key file field.	true

Table 4.3. Google Analytics Retrieval Configuration Options

Retrieval configuration should be applied in InternalAnalyticsSettings, see Internal Settings [25]. Example:

```
<StructProperty Name="googleAnalytics">
  <Struct>
    <StringProperty Name="applicationName">appName</StringProperty>
    <IntProperty Name="pid">12345678</IntProperty>
    <StringProperty
Name="serviceAccountEmail">ID@developer.gserviceaccount.com</StringProperty>

    <LinkProperty Name="p12File"
LinkType="coremedia:///cap/contenttype/CMDownload"
      xlink:href="Options/Settings/googleAnalytics%2Ep12.xml"
      cmexport:path="Settings/googleAnalytics%2Ep12"/>
  </Struct>
</StructProperty>
```

CAUTION

Please take care of security protection of the CMDownload content item containing your Google Analytics p12 Key File:

- Restrict read access to the CMDownload content item
- Ensure that no link is generated to the CMDownload content item, for example by linking to it in another content.
- Exclude the content item from website search by checking the corresponding option.
- Headless Server: Configure the document path in the `blocklist` to prevent delivery of the file in headless environments.

CAUTION

Please refer to the Google Analytics Developers Documentation for further information on request quotas.

4.2 Studio Extension

CoreMedia Studio plugins provide editing forms for the configuration of the "top-n-list" content items. They are available for the analytics content types `CMALXPageList` and `CMALXEventList`.

4.3 Analytics Feedback

To get direct feedback on the performance of an article, a page impression history is displayed in *CoreMedia Studio* on the metadata tab of each article. Furthermore, a *Site Performance Widget* can be configured on the *Dashboard* to show the total number of page impressions and publication events for a site.

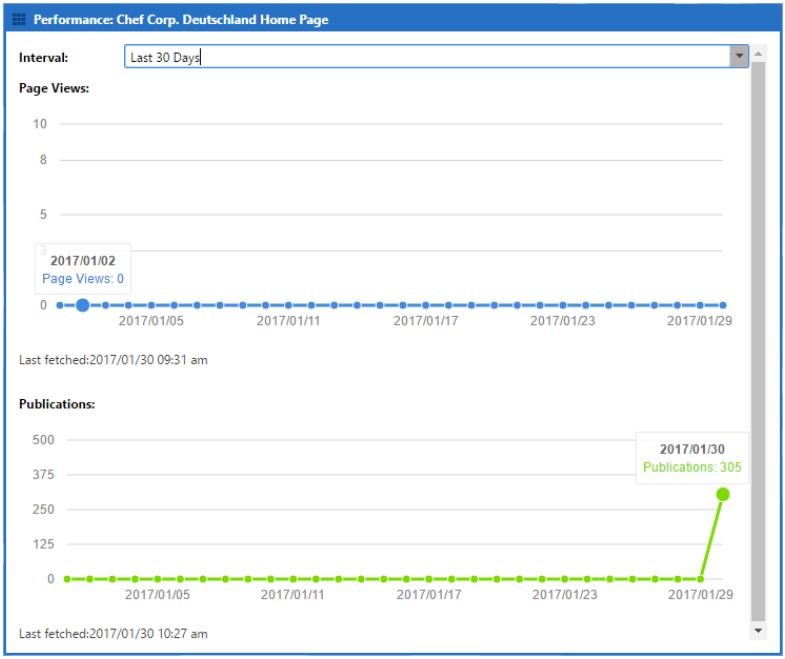

Figure 4.1. Page Impression History for 30 days

The page impression history can be displayed for the last 7 or 30 days.

As the page impressions are retrieved asynchronously from a third-party analytics provider, the timestamp of the last retrieval is displayed.

Data is currently retrieved from Google Analytics only. Corresponding configuration must be provided.

The configuration option `pageViewHistoryInterval` is used exclusively for the page impression history. The setting defines the interval for requests to the analytics

service provider. The configuration option `interval` has no effects for this functionality.

Technical Variable Name	Description/Value	Required	Default Value
`pageViewHistoryInterval`	The update interval for requests to the analytics provider in minutes.	false	1440 (1 day)
`publicationHistoryInterval`	The update interval for calculating the site publication history in minutes.	false	180 (3h)
`publicationHistoryDocumentType`	The content type used for calculating the site publication history. It may contain a comma separated list.	false	CMLinkable

Table 4.4. Google Analytics Retrieval Configuration Options

NOTE

The jobs for data retrieval is triggered each minute. The analytics retrieval task will only retrieve data from the analytics provider if the configuration of the analytics provider has changed, on the first run or after the default interval (24 h) or the customized interval `pageViewHistoryInterval` has expired.

Page impressions equal to zero represent either no page impressions or no data available. If no data is available for the selected time span, no chart is displayed.

The page view history data is cleared for a content item if the data has not been updated in the last 30 days.

The publication history will be calculated, when the content type configuration has changed or when the data is older than configured in the interval configuration.

Publication events are integrated into the page impression history for articles. When you hover over the publication history, the corresponding date is marked in the page impression history.

Publication events are also aggregated per site and displayed in the *Site Performance Widget*. Aggregation is performed asynchronously, a timestamp indicates the date of the last retrieval. The default interval for aggregation is three hours.

5. Integrating an Analytics Service Provider

To integrate another analytics service provider you have to consider the following aspects:

- Configure the tracking and retrieval information of the new provider on the root channel of a site using the generic struct editor.
- Create a tracking extension that hooks into *CoreMedia CAE*.
- Create a retrieval extension that hooks into `es-alx-retrieval-component`. The extension requires a single managed bean of type `AnalyticsServiceProvider`.

5.1 Studio Extension

Forms for configuring tracking can be hooked into the `CMALXBaseList` editing forms using a *CoreMedia Studio* extension. It is good practice to further extend the existing extension points:

- `CMALXBaseList` form

  The form contains a tab for each analytics provider, so add a new tab. The tab contains only the retrieval configuration fields.

For more information have a look at the module `alx-studio-plugin` as it provides the base components to be used to implement the Studio plugin for analytics service providers.

5.2 CAE Extension

The purpose of the CAE integration is to add the tracking aspect to page impressions and user interactions with your website.

The recommended way is to use a `ViewHookEventListener` that adds tracking to `com.coremedia.blueprint.common.contentbeans.Page` instances. Add an implementation of `AnalyticsProvider` and a template that should prepare tracking calls in its `head` view. These views are automatically included when rendering a page and your ViewHookEventListener is present.

5.3 Elastic Worker Extension

Set up an extension for the `es-alx-retrieval-component` providing a managed bean of type `AnalyticsServiceProvider`. Have a look at the implementation of `ElasticGoogleAnalyticsServiceProvider` as example for how to implement synchronous access.

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 *CAE Feeder* makes content beans searchable by sending their data to the *Search Engine*, which adds it to the index.
Content Application Engine (CAE)	The *Content Application Engine* (*CAE*) is a framework for developing content applications with *CoreMedia CMS*. 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 *CoreMedia CMS* 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 *Content Delivery Environment* is the environment in which the content is delivered to the end-user. It may contain any of the following modules: • *CoreMedia Master Live Server* • *CoreMedia Replication Live Server* • *CoreMedia Content Application Engine* • *CoreMedia Search Engine* • *Elastic Social* • *CoreMedia Adaptive Personalization*

Content Feeder	The *Content Feeder* is a separate web application that feeds content items of the CoreMedia repository into the *CoreMedia Search Engine*. Editors can use the *Search Engine* to make a full text search for these fed items.
Content item	In *CoreMedia CMS*, 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 *Content Management Environment* is the environment for editors. The content is not visible to the end user. It may consist of the following modules: • *CoreMedia Content Management Server* • *CoreMedia Workflow Server* • *CoreMedia Importer* • *CoreMedia Site Manager* • *CoreMedia Studio* • *CoreMedia Search Engine* • *CoreMedia Adaptive Personalization* • *CoreMedia Preview CAE*
Content Management Server	Server on which the content is edited. Edited content is published to the Master Live Server.
Content Repository	*CoreMedia CMS* 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	*Content Server* is the umbrella term for all servers that directly access the CoreMedia repository: *Content Servers* are web applications running in a servlet container. • *Content Management Server* • *Master Live Server* • *Replication Live Server*
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 *CoreMedia CMS*. They are written by CoreMedia developers - be it clients, partners or CoreMedia employees. CoreMedia contributions are hosted on Github at https://github.com/coremedia-contributions.
Control Room	*Control Room* is a *Studio* plugin, which enables users to manage projects, work with workflows, and collaborate by sharing content with other *Studio* users.
CORBA [Common Object Request Broker Architecture]	The term *CORBA* 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	*CoreMedia Studio* 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, *CoreMedia Studio* 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: `<!DOCTYPE coremedia SYSTEM "http://www.coremedia.com/dtd/coremedia.dtd"` 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	*CoreMedia Elastic Social* is a component of *CoreMedia CMS* that lets users engage with your website. It supports features like comments, rating, likings on your website. *Elastic Social* is integrated into *CoreMedia Studio* 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.
Headless Server	CoreMedia Headless Server is a CoreMedia component introduced with CoreMedia Content Cloud which allows access to CoreMedia content as JSON through a GraphQL endpoint. The generic API allows customers to use CoreMedia CMS for headless use cases, for example delivery of pure content to Native Mobile Applications, Smart-

	watches/Wearable Devices, Out-of-Home or In-Store Displays or Internet-of-Things use cases.
Home Page	The main entry point for all visitors of a site. Technically it is often referred to as root document and also serves as provider of the default layout for all subpages.
IETF BCP 47	Document series of *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.
Importer	Component of the CoreMedia system for importing external content of varying format.
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 ActionScript/MXML/Maven based version for CMCC 10. Since CMCC 11 (2110), Jangaroo uses TypeScript and is implemented as a *Node.js* and *npm* 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.
JSP	JSP (Java Server Pages) is a template technology based on Java for generating dynamic HTML pages. It consists of HTML code fragments in which Java code can be embedded.
Locale	Locale is a combination of country and language. Thus, it refers to translation as well as to localization. Locales used in translation processes are typically represented as IETF BCP 47 language tags.
Master Live Server	The *Master Live Server* is the heart of the *Content Delivery Environment*. It receives the published content from the *Content Management Server* and makes it available to the *CAE*. If you are using the *CoreMedia Multi-Master Management Extension* you may use multiple *Master Live Server* 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.
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 *Replication Live Server* is to distribute load on different servers and to improve the robustness of the *Content Delivery Environment*. The *Replication Live Server* is a complete Content Server installation. Its content is an replicated image of the content of a *Master Live Server*. The *Replication Live Server* updates its database due to change events from the *Master Live Server*. You can connect an arbitrary number of *Replication Live Servers* to the *Master Live Server*.
Resource	A folder or a content item in the CoreMedia system.
ResourceURI	A ResourceUri uniquely identifies a page which has been or will be created by the *Active Delivery Server*. The ResourceUri consists of five components: Resource ID, Template ID, Version number, Property names and a number of key/value pairs as additional parameters.
Responsive Design	Responsive design is an approach to design a website that provides an optimal viewing experience on different devices, such as PC, tablet, mobile phone.
Site	A site is a cohesive collection of web pages in a single locale, sometimes referred to as localized site. In *CoreMedia CMS* 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 `CMSite`.
Site Manager	Swing component of CoreMedia for editing content items, managing users and workflows. The Site Manager is deprecated for editorial use.
Site Manager Group	Members of a site manager group are typically responsible for one localized site. Responsible means that they take care of the contents of that site and that they accept translation tasks for that site.
Template	In CoreMedia, JSPs used for displaying content are known as Templates. OR In *Blueprint* a template is a predeveloped content structure for pages. Defined by typically an administrative user a content editor can use this template to quickly create a complete new page including, for example, navigation, predefined layout and even predefined content.
Translation Manager Role	Editors in the translation manager role are in charge of triggering translation workflows for sites.
User Changes web application	The *User Changes* web application is a *Content Repository* listener, which collects all content, modified by *Studio* users. This content can then be managed in the *Control Room*, as a part of projects and workflows.
Variants	Most of the time used in context of content variants, variants refer to all localized versions within the complete hierarchy of master and their derived sites (including the root master itself).
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 *CoreMedia CMS* always guarantees link consistency. But links can be declared with the *weak* 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 and global-search-and-replace but also executes freely definable workflows.
XLIFF	XLIFF is an XML-based format, standardized by OASIS for the exchange of localizable data. An XLIFF file contains not only the text to be translated but also metadata about the text. For example, the source and target language. *CoreMedia Studio* allows you to export content items in the XLIFF format and to import the files again after translation.

Index