



# Solution Overview for Business Users

CoreMedia Experience Platform – 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.

January 28, 2026 (Release 2512.0 )



1. Overview of CoreMedia Experience Platform .....	1
2. Overview of Brand Blueprint .....	3
3. Overview of eCommerce Blueprint .....	6
3.1. CoreMedia Content Cloud for Salesforce Commerce Cloud .....	9
3.1.1. Screenshots of the Integration – the Business User View .....	9
3.1.2. Adding CMS Fragments to Shop Pages .....	23
3.1.3. CoreMedia Content Widget .....	24
3.1.4. Studio Integration of Commerce Content .....	25
3.2. CoreMedia Content Cloud for SAP Hybris .....	31
3.2.1. Screenshots of the Integration – the Business User View .....	31
3.2.2. Adding CMS Fragments to Shop Pages .....	45
3.2.3. CoreMedia Content Widget .....	47
3.2.4. Studio Integration of Commerce Content .....	48
3.3. CoreMedia Content Cloud for commercetools .....	55
3.4. Custom Commerce Adapter .....	59
3.4.1. Commerce Hub Architecture .....	59
3.4.2. Commerce Hub API .....	60
3.4.3. Integrating a Custom Commerce System .....	62
4. Asset Management .....	67
4.1. Multiple Images and Sizes .....	68
4.2. Working with Assets .....	70
4.2.1. Extracted Metadata .....	71
4.2.2. Creating Assets in Studio .....	73
4.2.3. Creating Pictures from Assets .....	75
4.2.4. Creating Videos from Assets .....	77
4.2.5. Categorizing Assets .....	78
4.2.6. Searching for Assets .....	80
4.2.7. Publishing Assets .....	81
4.2.8. Configuring the Asset Download Portal .....	82
4.3. Editing Images .....	84
4.4. Editing Image Maps .....	93
4.5. Advanced Teaser Management .....	95
4.5.1. Positioning Text on Teasers .....	95
4.5.2. Creating Styles for Teaser Overlays .....	97
4.5.3. Setting a Call-to-action Button .....	97
4.6. Editing 360°-Views .....	100
4.7. Editing Shoppable Videos .....	101
5. Translation, Localization, Globalization .....	103
5.1. Overview .....	104
5.1.1. Designing Your Multi-Site Experience .....	105
5.2. Terms .....	112
5.3. Sites Structure .....	116
5.4. Translating Content .....	121
5.4.1. Preparing Translation: Deriving a Translated Site .....	122
5.4.2. Preparing Translation: Find Content Items That Need Translation .....	124

5.4.3. Performing Translation: Start, Finish and Abort .....	126
5.5. Synchronizing Content .....	134
5.5.1. Deriving a Synchronized Site .....	134
5.5.2. Synchronizing Changes between Master and Derived Site .....	137
5.5.3. Removing Content Permanently from Synchronization .....	140
6. Workflow Management .....	176
6.1. Workflow App .....	143
6.2. Publication .....	146
6.2.1. Approval and Publication of Folders and Content Items .....	146
6.2.2. Predefined Publication Workflows .....	148
6.2.3. Features of the Publication Workflows .....	148
6.3. Translation Workflow .....	150
6.3.1. Roles and Rights .....	150
6.3.2. Configuration and Customization .....	151
7. Basic Content Management .....	152
7.1. Content Type Model .....	153
7.1.1. Common Content Types .....	153
7.1.2. ....	154
7.1.3. Content Type Sitemap .....	154
7.2. Folder and User Rights Concept .....	155
7.3. Navigation and Contexts .....	157
7.4. Settings .....	160
7.5. Page Assembly .....	163
7.6. Adding Comments on Content Properties .....	167
7.7. Setting a Displayed Date .....	169
7.8. Time Dependent Visibility .....	170
7.9. Checking Content in External Preview .....	173
7.10. Working with Authors .....	175
7.11. Versioning and Lifecycle .....	176
7.12. Validators .....	178
7.13. Getting Keyword Recommendations .....	180
7.14. Tagging and Taxonomies .....	182
7.14.1. Taxonomy Management .....	183
7.14.2. Taxonomy Assignment .....	184
7.14.3. Metadata Management .....	186
7.15. Content Lists .....	193
7.16. View Types .....	196
7.17. Dynamic Templating .....	198
7.18. Managing End User Interactions .....	199
7.19. URLs .....	200
7.20. Vanity URLs .....	201
7.21. Content Visibility .....	202
7.22. Robots File .....	203
7.23. Sitemap .....	204
7.24. Website Search .....	205
7.25. ....	206

8. Managing Users and Groups .....	206
8.1. Opening the User Manager .....	207
8.2. Searching for Users and Groups .....	208
8.3. Creating a New User .....	209
8.4. Creating a New Group .....	212
8.5. Adding, Deleting and Editing Rules .....	216
8.6. Defining Rights on Folders .....	218
9. CoreMedia Content as a Service (CaaS) .....	220
9.1. Endpoints of the <i>Headless Server</i> .....	222
9.2. Preview .....	224
9.2.1. JSON Preview Client .....	224
9.2.2. Custom Preview Client .....	224
9.3. Security .....	225
9.3.1. Query Allow List for GraphQL Queries .....	226
9.3.2. Limiting the Size of a Search Result .....	226
9.3.3. Limiting the Depth of a GraphQL Query .....	227
9.3.4. Limiting the Complexity of a GraphQL Query .....	227
9.3.5. Enforcing an Execution Timeout for GraphQL Queries .....	228
9.3.6. MediaType Content Negotiation .....	228
9.4. Search .....	230
9.4.1. Generic Search .....	230
9.4.2. Dynamic Query Lists .....	237
9.4.3. Custom Filter Queries .....	238
9.5. Rich Text .....	242
9.5.1. Rich Text Output .....	242
9.5.2. Using RichTextAdapters for Different Rich Text Grammars .....	244
9.6. eCommerce Extension .....	245
9.6.1. Headless Commerce Integration Architecture .....	246
9.6.2. Augmentation .....	247
9.6.3. Product Lists .....	252
9.6.4. References to Products and Categories .....	253
9.6.5. eCommerce Setup and Configuration .....	254
9.7. Persisted Queries .....	256
9.7.1. Loading Persisted Queries at Server Startup .....	257
9.7.2. Query Allow Listing .....	259
9.7.3. Apollo Automatic Persisted Queries .....	260
9.8. Site Filter .....	261
9.9. Media Endpoint .....	263
9.9.1. Media Endpoint URLs .....	264
9.10. REST Access to GraphQL .....	267
9.10.1. Mapping REST Access to Persisted Queries .....	268
9.10.2. JSLT Transformation .....	269
9.11. Metadata Root .....	271
9.11.1. PDE Mapping as Metadata .....	271
9.12. ....	274
9.13. Frontend Client Development .....	274
9.13.1. Rendering the Homepage of a Site .....	274

10. CoreMedia Campaigns .....	278
10.1. CoreMedia Campaigns Overview .....	279
10.1.1. Visibility of Campaigns .....	279
10.1.2. Characteristics of Campaign Content .....	279
10.1.3. What is CoreMedia Campaigns? .....	280
10.1.4. Features .....	280
10.1.5. Getting the Content with GraphQL .....	281
10.1.6. Overview of the Architecture .....	283
10.1.7. Campaign Delivery .....	283
10.2. Create a Campaign .....	284
11. CoreMedia Event Hub Service .....	291
11.1. Introduction to the Event Hub Service .....	292
11.1.1. Events .....	292
11.1.2. Webhooks .....	293
11.2. Working with the Event Hub Service .....	295
11.2.1. Prerequisites .....	295
11.2.2. Create New Webhook Subscription .....	295
11.2.3. Read Existing Webhook Subscriptions .....	298
11.2.4. Update Existing Webhook Subscriptions .....	299
11.2.5. Delete Existing Webhook Subscription .....	299
11.2.6. Table of Error Codes and Messages .....	300
12. Image Transformation Service .....	302
13. CoreMedia Ingest Service .....	304
14. Integration: CoreMedia Hubs and Connectors .....	310
14.1. CoreMedia Content Hub .....	312
14.2. CoreMedia Commerce Hub .....	313
14.3. CoreMedia Feedback Hub .....	314
15. CoreMedia Blueprint 3rd party Integrations .....	315
15.1. Salesforce Marketing Cloud Integration .....	316
15.2. Open Street Map Integration .....	318
16. CoreMedia Labs .....	319
16.1. Feedback Hub Adapter – AB/n Testing .....	320
16.2. Feedback Hub Adapter – Acrolinx .....	324
16.3. Feedback Hub Adapter – Searchmetrics .....	326
16.4. Feedback Hub Adapter – Siteimprove .....	330
16.5. Feedback Hub Adapter – OpenAI ChatGPT .....	332
16.6. Feedback Hub Adapter – wonk.ai .....	334
16.7. CoreMedia Content Hub Adapter – Dropbox .....	337
16.7.1. Configuring the content-hub-adapter-dropbox .....	338
16.7.2. Usage .....	340
16.8. CoreMedia Content Hub Adapter – Cloudinary .....	342
16.8.1. Configuring the content-hub-adapter-cloudinary .....	342
16.8.2. Usage .....	345
16.9. CoreMedia Content Hub Adapter – CoreMedia Repository .....	347
16.10. Social Media Hub .....	349
16.11. Translations with GlobalLink Connect Cloud .....	351
16.12. Monetate Integration .....	356

- 16.13. Spark – The React Example Application ..... 360
- 16.14. Scheduled Publication Workflow ..... 362
- 16.15. Three-Step Publication Workflow ..... 366
- 17. CoreMedia Engagement Cloud ..... 373
  - 17.1. CoreMedia Engagement Cloud Integrations ..... 376
    - 17.1.1. What is an Integration? ..... 376
    - 17.1.2. Why Integrating with External Entities or Pro-  
viders? ..... 377
    - 17.1.3. CoreMedia Integrations ..... 378
    - 17.1.4. CoreMedia REST API ..... 380
    - 17.1.5. Javascript API ..... 380
    - 17.1.6. CoreMedia SFTP Service ..... 381
    - 17.1.7. CoreMedia Integration App ..... 382
    - 17.1.8. CoreMedia Export App ..... 388
    - 17.1.9. CoreMedia Custom App ..... 389
    - 17.1.10. CoreMedia DataSources ..... 389
  - 17.2. CoreMedia Tag and Cookies ..... 391
    - 17.2.1. CoreMedia Tag ..... 391
    - 17.2.2. CoreMedia Cookies ..... 392
    - 17.2.3. CoreMedia Service ..... 393
    - 17.2.4. CoreMedia Campaigns and Tag Management ..... 395
    - 17.2.5. CoreMedia and Third-party Cookies Manage-  
ment ..... 397
- 18. Documentation and Training ..... 399

## List of Figures

2.1. The Brand Blueprint in CoreMedia Studio .....	9
2.2. Side-by-Side Translation/Review with Highlighted Changes .....	11
2.3. Configure your Personal Dashboard .....	12
2.4. Integration of Single-Page-Applications (Headless Mode) .....	23
3.1. The integration of CoreMedia with Salesforce Commerce Cloud in CoreMedia Studio .....	7
3.2. The integration of CoreMedia with SAP Hybris in CoreMedia Studio .....	7
3.3. The integration of CoreMedia with a headless commerce system (content-led integration scenario) .....	8
3.4. Using CoreMedia Studio to position inspirational content within Salesforce Commerce Cloud .....	9
3.5. Responsive Preview of the storefront experience within CoreMedia Studio (here: tablet view) .....	10
3.6. Responsive Preview of the storefront experience within CoreMedia Studio (here: mobile view) .....	23
3.7. Image Cropping within CoreMedia Studio .....	23
3.8. Adding inspirational content to Category/Listing Pages .....	11
3.9. Managing the Navigation Menu within CoreMedia Studio .....	12
3.10. Product Catalog Access within CoreMedia Studio – incl. Product Variant Support .....	23
3.11. Embedding Product and Category Banners within Stories .....	23
3.12. Enriching Standard Product Teasers .....	23
3.13. Shoppable Images – Creating Hot Zones within Images .....	23
3.14. Advanced Text on Image Formatting and Positioning .....	23
3.15. Creating content-driven landing pages within the Salesforce store- front .....	23
3.16. Advanced Tagging (incl. the option to connect to visual recognition services based on AI) .....	23
3.17. Scheduling Campaigns .....	23
3.18. Time travel preview for scheduled campaigns .....	23
3.19. SEO – Human readable URLs and advanced metadata editing .....	23
3.20. Non-product search .....	23
3.21. PDP Augmentation – Alternative Images, Videos, Downloads .....	23
3.22. PDP Augmentation – Inspirational Content .....	23
3.23. 360° Product Spinners .....	23
3.24. Content reuse for Newsletters / Outbound Marketing .....	23
3.25. Content differencing .....	23
3.26. Reviewed Publication Workflows .....	23
3.27. Collaboration in Teams .....	23
3.28. Personalization based on click behavior or customer groups within Salesforce .....	23
3.29. Persona-based Preview of the Experience .....	23
3.30. Dashboard Overview .....	23
3.31. Strong Translation / Localization Capabilities .....	23
3.32. The integration of CoreMedia with a headless commerce system (content-led integration scenario) .....	25

3.33. Library with catalog in the tree view .....	26
3.34. Library tree with multiple occurrences of the same category .....	27
3.35. Test Customer Persona with Commerce Customer Segments .....	53
3.36. Edit Commerce Segments in Test Customer Persona .....	29
3.37. Choosing a page layout for a shop page .....	30
3.38. SAP Hybris example .....	31
3.39. Using CoreMedia Studio to position inspirational content within SAP Hybris .....	32
3.40. Responsive Preview of the storefront experience within CoreMedia Studio (here: tablet view) .....	32
3.41. Responsive Preview of the storefront experience within CoreMedia Studio (here: mobile view) .....	33
3.42. Image Cropping within CoreMedia Studio .....	33
3.43. Adding inspirational content to Category/Listing Pages .....	34
3.44. Managing the Navigation Menu within CoreMedia Studio .....	34
3.45. Product Catalog Access within CoreMedia Studio – incl. Product Variant Support .....	35
3.46. Embedding Product and Category Banners within Stories .....	35
3.47. Enriching Standard Product Teasers .....	36
3.48. Shoppable Images – Creating Hot Zones within Images .....	36
3.49. Advanced Text on Image Formatting and Positioning .....	37
3.50. Creating content-driven landing pages .....	37
3.51. Advanced Tagging (incl. the option to connect to visual recognition services based on AI) .....	38
3.52. Rule-Driven Content Lists – Dynamic Lists .....	38
3.53. Scheduling Campaigns .....	39
3.54. Time travel preview for scheduled campaigns .....	39
3.55. SEO – Human readable URLs and advanced metadata editing .....	40
3.56. PDP Augmentation – Alternative Images, Videos, Downloads .....	40
3.57. PDP Augmentation – Inspirational Content .....	41
3.58. 360° Product Spinners .....	41
3.59. Content reuse for Newsletters / Outbound Marketing .....	42
3.60. Content differencing .....	42
3.61. Reviewed Publication Workflows .....	43
3.62. Collaboration in Teams .....	43
3.63. Personalization based on click behaviour or customer segments within SAP Hybris .....	44
3.64. Dashboard Overview .....	44
3.65. Strong Translation / Localization Capabilities .....	45
3.66. Various Shop Pages with CMS Fragments .....	46
3.67. Using the <i>CoreMedia Content Widget</i> – A Homepage Frag- ment .....	47
3.68. Using the <i>CoreMedia Content Widget</i> – Connection to CMS Content via placement name .....	48
3.69. Library with catalog in the tree view .....	49
3.70. Library tree with multiple occurrences of the same category .....	49
3.71. Open Product in tab .....	50
3.72. Product in tab preview .....	50
3.73. Open Category in tab .....	51

3.74. Category in tab preview .....	51
3.75. Test Customer Persona with Commerce Customer Segments .....	53
3.76. Edit Commerce Segments in Test Customer Persona .....	54
3.77. Library with catalog in the tree view .....	55
3.78. Library tree with multiple occurrences of the same category .....	56
3.79. Open Product in tab .....	57
3.80. Product in tab with JSON preview .....	57
3.81. Open Category in tab .....	58
3.82. Architectural overview of the Commerce Hub .....	59
3.83. More detailed architecture view .....	60
4.1. Picture Asset content item .....	71
4.2. IPTC metadata setting in Photoshop .....	73
4.3. Metadata for Assets .....	74
4.4. Create picture from asset button .....	75
4.5. Newly created picture shown in Picture Asset .....	76
4.6. Product linked by picture .....	76
4.7. Create video from asset button .....	77
4.8. Newly created video shown in Video Asset .....	78
4.9. Taxonomy for assets .....	78
4.10. Add category to asset .....	79
4.11. Download portal with Asset .....	80
4.12. Select property for publication .....	81
4.13. Asset download portal .....	82
4.14. Configuration of the download portal .....	83
4.15. Cropping images .....	84
4.16. Overview with all selection frames .....	85
4.17. Moved focus point with automatically arranged 1:1 crop .....	86
4.18. Decoupled selection frame .....	86
4.19. The crop is too small .....	87
4.20. Enlarged image with white background .....	88
4.21. Move focus point .....	89
4.22. Effect of rotation on the selection frames .....	90
4.23. Exposure controls .....	91
4.24. Undo steps .....	92
4.25. Hot Zones in an image map (pop-up) .....	93
4.26. Configuring image maps .....	94
4.27. Open Teaser panel .....	95
4.28. Select style for movable teaser text .....	96
4.29. Formatting text .....	96
4.30. Position text .....	97
4.31. Call-to-action button in teaser view .....	98
4.32. Call-to-action checkbox in teasable content item .....	98
4.33. Call-to-action checkbox in Teaser content item .....	98
4.34. Images of a dress with a slight counterclockwise rotation .....	100
4.35. The shoppable video editor .....	101
4.36. Site preview with shoppable video .....	102
5.1. Side-by-side translation (review) within CoreMedia Studio .....	103
5.2. Master content and Derived content .....	104
5.3. Sites: Language First .....	107



5.4. Sites: Country First .....	108
5.5. Derived Sites within CoreMedia .....	116
5.6. Multi-Site Interdependence .....	118
5.7. Locales Administration in CoreMedia Studio .....	119
5.8. Derive Site: Setting site manager group .....	120
5.9. Sites Window .....	122
5.10. Derive a site dialog .....	123
5.11. Derived Site in Sites Window .....	124
5.12. Translation State Dashboard Widget .....	125
5.13. Translation State Library Filter .....	125
5.14. Translation workflow detail panel next steps .....	127
5.15. Download XLIFF file .....	128
5.16. Translation Workflow Window .....	129
5.17. Translation workflow panel .....	131
5.18. Aborting translation workflow in Control Room .....	132
5.19. Aborting translation workflow in Workflow App .....	133
5.20. Sites Window .....	135
5.21. Derive a site dialog .....	136
5.22. Derived synchronized sites .....	137
5.23. Localization Workflow Window .....	138
5.24. Warning for deselected subsites .....	139
5.25. Uncheck the synchronization checkbox .....	141
6.1. Starting Workflow App from the Main Menu of Studio .....	143
6.2. Workflow app main view .....	143
6.3. Opened workflow in Workflow app .....	144
6.4. Workflow App in split-screen mode with Content App .....	145
6.5. Starting a Publication Workflow .....	146
6.6. Starting a translation workflow .....	150
7.1. Content Type Sitemap .....	154
7.2. Navigation in the Site .....	157
7.3. Navigation children .....	159
7.4. Struct editor in a Settings content item .....	161
7.5. The page grid editor .....	165
7.6. Editorial Comments in the Content Form .....	167
7.7. Editorial Comments in the Feedback Hub .....	168
7.8. Setting a Displayed Date .....	169
7.9. The Validity field .....	170
7.10. Icons for validity .....	171
7.11. The visibility .....	171
7.12. Time Travel dialog .....	172
7.13. Preview dialog .....	173
7.14. External Preview menu .....	173
7.15. Authors .....	175
7.16. Comparing versions .....	176
7.17. Validators in the Form .....	178
7.18. Searching for issues in content .....	179
7.19. Feedback Hub Window with keywords from Imagga system .....	180
7.20. Nagbar showing Error during the loading of keywords .....	181
7.21. Dynamic Query List with taxonomy condition: Black .....	182

7.22. Dynamic list of articles tagged with "Black" .....	182
7.23. Taxonomy Administration Editor .....	184
7.24. Taxonomy Property Editor .....	185
7.25. Taxonomy Studio Settings .....	185
7.26. Taxonomy Localization Form .....	186
7.27. Query List with taxonomy condition .....	187
7.28. Categorized Article .....	188
7.29. Tag chooser .....	188
7.30. Taxonomy filter in the Library .....	190
7.31. The taxonomy editor .....	191
7.32. Adding a new tag .....	191
7.33. Check for linking content items .....	192
7.34. A standard collection .....	193
7.35. Dynamic Query List with taxonomy condition .....	194
7.36. Dynamic Elastic Social List .....	195
7.37. Layout Variant selector .....	197
7.38. The Validity field .....	202
7.39. TimeTravelDialog .....	202
7.40. Channel settings with configuration for Robots.txt as a linked setting on a root page .....	203
8.1. Open User Manager .....	207
8.2. Search fields .....	208
8.3. Create new user with icon .....	209
8.4. Create new user popup windows .....	210
8.5. Set home folder and group membership .....	210
8.6. Check effective rules .....	211
8.7. Create new group .....	212
8.8. Create a new group popup windows .....	213
8.9. Set group members and membership .....	214
8.10. Check effective rules .....	215
8.11. Adding rules .....	216
8.12. New rule in list .....	216
8.13. Open the Properties window .....	218
8.14. Add rules to folder .....	219
9.1. Headless Server overview .....	220
9.2. Headless Commerce Integration Example .....	246
9.3. Headless server request/response flow using REST .....	268
9.4. Screenshot of the example homepage .....	277
10.1. Example campaign with defined refinements .....	281
10.2. Only the summer collection Campaigns are retrieved for the site with the id "ced8921..." .....	282
10.3. Overview of the architecture .....	283
10.4. Start Campaign Creation from Project .....	284
10.5. Start Campaign Creation from the Main Menu .....	285
10.6. Create New Campaigns Dialog .....	285
10.7. Campaigns with Create Campaign Button .....	286
10.8. Enter start and end date of campaign .....	286
10.9. Campaign Details .....	287
10.10. Assign content .....	287

10.11. Prioritize and Start your Campaign .....	289
10.12. Running Campaign .....	290
11.1. Architecture of Event Hub Service .....	291
13.1. Import Sequence .....	388
13.2. Sequence for asynchronous bulk requests .....	307
14.1. Use Cases .....	310
14.2. Connector for Marketing Cloud .....	311
14.3. Import content from RSS feed .....	312
14.4. Example of Imagga Integration .....	314
15.1. Push content to Marketing Cloud .....	316
15.2. SFMC Journey condition example .....	317
15.3. Adding SFMC Journeys in personalization preview personas .....	317
15.4. Example for an Open Street Map integration in a website .....	318
16.1. Teaser for A/B testing .....	321
16.2. Create A/B/n test in Studio .....	322
16.3. A/B/n test results .....	323
16.4. Acrolinx window .....	324
16.5. Feedback for CoreMedia richtext .....	325
16.6. Overall Scoring Overview .....	326
16.7. Keyword Scoring .....	327
16.8. Content Questions .....	328
16.9. Competitors Overview .....	329
16.10. Siteimprove Site Setup .....	330
16.11. Siteimprove Feedback shown in Studio .....	331
16.12. ChatGPT Integration in Studio .....	333
16.13. Write integration .....	334
16.14. Optimize teaser text .....	335
16.15. Optimize Keywords .....	336
16.16. Content Hub adapters in Library .....	337
16.17. Full configuration of adapter in global space .....	340
16.18. Configured Dropbox in Studio Library .....	341
16.19. Importing content from Dropbox in Studio Library .....	341
16.20. Cloudinary adapter in the Library .....	342
16.21. Full configuration of adapter in global space .....	345
16.22. Cloudinary adapter in the Library .....	345
16.23. Creating content from Cloudinary items .....	346
16.24. Repository adapter in Library .....	347
16.25. CoreMedia Social Media Hub .....	349
16.26. Start localization workflow .....	352
16.27. Configure workflow .....	352
16.28. Pending workflow is shown .....	353
16.29. Error handling .....	353
16.30. Translation is finished .....	354
16.31. Connection to GlobalLink Connect Cloud .....	355
16.32. Content for different customers .....	357
16.33. A/B testing .....	358
16.34. Teaser for A/B testing .....	359
16.35. Teaser for A/B testing .....	360
16.36. Start Publication Workflow .....	362

- 16.37. Select scheduled date ..... 363
- 16.38. Pending scheduled workflow ..... 363
- 16.39. Checking the workflow ..... 364
- 16.40. Completed workflow ..... 365
- 16.41. Start Publication Workflow ..... 366
- 16.42. Start Three-Step Publication Workflow ..... 367
- 16.43. Assign workflow for approving ..... 367
- 16.44. Pending workflow ..... 368
- 16.45. Details of workflow after start ..... 368
- 16.46. Workflow in inbox for review ..... 369
- 16.47. Details of the workflow before review ..... 369
- 16.48. Assign workflow to people ..... 370
- 16.49. Workflow in the inbox ..... 370
- 16.50. Details of the workflow before publication ..... 371
- 16.51. The publish step ..... 372
- 17.1. Engagement Studio ..... 388
- 17.2. Contact Center and Analytics Studio ..... 388
- 17.3. Contact Center and Analytics Studio ..... 388
- 17.4. CoreMedia Integration App ..... 388
- 17.5. CoreMedia Export App ..... 388
- 17.6. .... 395
- 17.7. .... 396
- 17.8. .... 397
- 17.9. .... 398
- 17.10. .... 398

## List of Tables

- 5.1. Icons for Translation ..... 121
- 5.2. Icons for Synchronization ..... 134
- 6.1. User options. .... 149
- 7.1. Time dependent visibility icon ..... 172
- 7.2. Taxonomy icons ..... 187
- 11.1. Table of error codes and Messages ..... 300
- 12.1. Sizes, formats and operations ..... 302
- 16.1. Toplevel entry ..... 338
- 16.2. Entries for all connectors ..... 339
- 16.3. Entries in settings struct ..... 339
- 16.4. Toplevel entry ..... 343
- 16.5. Entries for all connectors ..... 343
- 16.6. Entries in settings struct ..... 344
- 16.7. Toplevel entry ..... 348
- 16.8. Entries for all connectors ..... 348
- 16.9. Supported Networks ..... 350
- 17.1. Table of error codes and Messages ..... 393

## List of Examples

- 5.1. Multi-Site Folder Structure Example ..... 117
- 5.2. Site Folder Structure Example ..... 117
- 9.1. Configuring Content Type Resolution for PDF and EPS Files ..... 228
- 9.2. Example implementation of a custom filter query. .... 239
- 9.3. Retrieving the URI template of a picture ..... 263
- 9.4. Retrieving the URI template of a picture with an alternative image  
format ..... 263
- 9.5. Retrieving the URI or the fully qualified URL of the original file of a  
picture ..... 263
- 9.6. Page query with siteID ..... 274
- 9.7. Page Component render function ..... 275
- 9.8. Iterating over all rows of the PageGrid ..... 275
- 9.9. The PageGridPlacement Component ..... 276
- 17.1. Appending information ..... 381
- 17.2. .... 394
- 17.3. .... 394

# 1. Overview of CoreMedia Experience Platform

This document is intended for architects and developers who want to work with CoreMedia Experience Platform or who want to learn about the concepts of the product.

CoreMedia Experience Platform combines CoreMedia Content Cloud and CoreMedia Experience Cloud.

## CoreMedia Content Cloud

*CoreMedia Content Cloud* is the next-generation experience management platform from CoreMedia that lets you build highly engaging, multi-channel branded eCommerce experiences as well as corporate sites for your global customers.

Now, you can easily bridge the gap between a pure eCommerce system which is focused on the more transactional aspects of the buying process and content-driven brand sites that focus on engaging user experiences.

*CoreMedia Content Cloud* was designed to empower your team in creating and managing highly relevant and engaging experiences for your customers from a single, easy-to-use business user interface. Customers should always get the information they need, independent of the device they use or the time they connect – delivered in an optimized fashion for the current customer's context.

*CoreMedia Studio* allows business users to create and manage experiences based on context and to define and test rules and customer segments for personalization in real-time. Content can be easily mixed with catalog items. Editors can intuitively select the products and categories from the catalog and place them on the site just as they are accustomed from other web content.

*CoreMedia Content Cloud* ships with CoreMedia Blueprints for eCommerce and corporate sites that provide a high-level of prefabrication of common features and use cases. The source code is provided for easy customization to your specific needs for competitive differentiation.

Built upon industry-leading best practices with a fully responsive and adaptive mobile first design plus a wealth of ready-to-use layout modules, your develop-

ment team can jump start on a strong foundation proven in many customer projects whilst retaining full flexibility. A predefined Maven based development environment is provided.

*Headless Server* allows you using *CoreMedia Experience Platform* in a headless way. *Headless Server* delivers content from the repository as JSON data over a GraphQL endpoint and is fully integrated in *CoreMedia Studio*. Therefore, you can preview and edit all changes in Studio.

CoreMedia Content Cloud bundles all components to help you manage every aspect of your blended digital experiences from content to commerce – regardless of the technical integration being commerce-led, content-led, hybrid or headless.

## Coremedia Engagement Cloud

CoreMedia Engagement Cloud is a powerful platform designed to enhance digital marketing efforts through advanced personalization and customer engagement tools. It integrates seamlessly with various marketing technologies, allowing businesses to deliver targeted and relevant content across multiple channels. With its robust analytics and segmentation capabilities, CoreMedia Engagement Cloud helps organizations optimize their content strategies and improve customer experiences.



## 2. Overview of Brand Blueprint

The *Brand Blueprint* provides a modern, appealing, highly visual website template that can be used to start a customization project. It demonstrates the capability to build localizable, multi-national, non-commerce web sites.

Based on a fully responsive, mobile-first design paradigm, the *Brand Blueprint* leverages most of our bricks and Design framework for easy customization and adaptation by frontend developers. It is a child theme, inherited from the Shared-Example Theme.

It scales from mobile via tablet to desktop viewport sizes and uses the CoreMedia Adaptive and Responsive Image Framework to dynamically deliver the right image sizes in the right aspect ratios and crops.

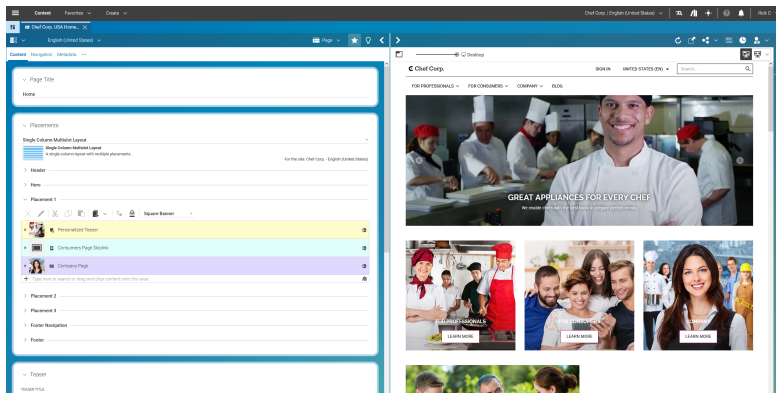


Figure 2.1. The Brand Blueprint in CoreMedia Studio

Overview of Brand Blueprint |

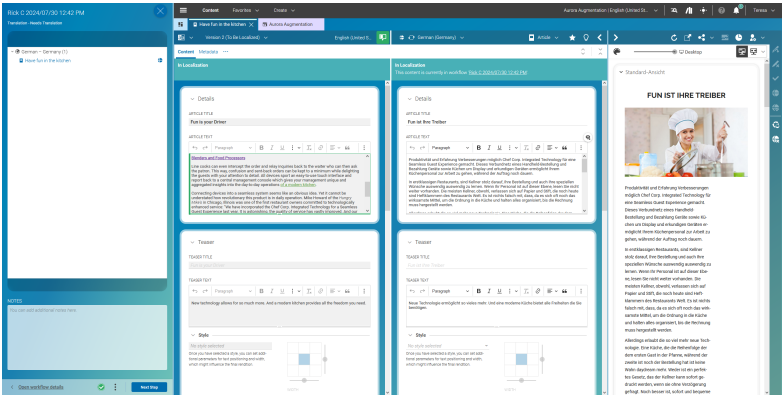


Figure 2.2. Side-by-Side Translation/Review with Highlighted Changes

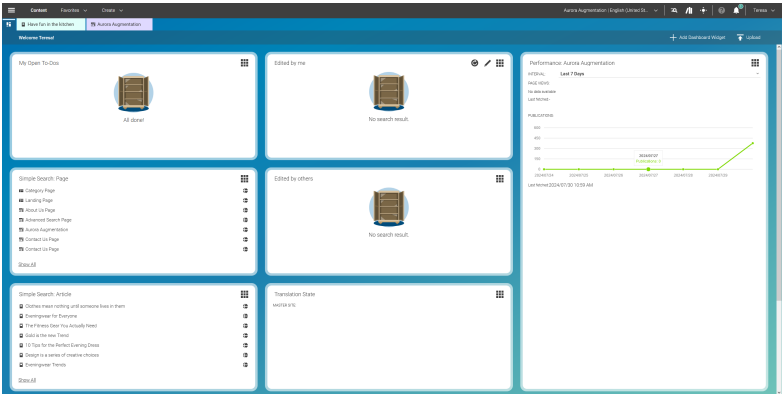


Figure 2.3. Configure your Personal Dashboard

# Overview of Brand Blueprint |

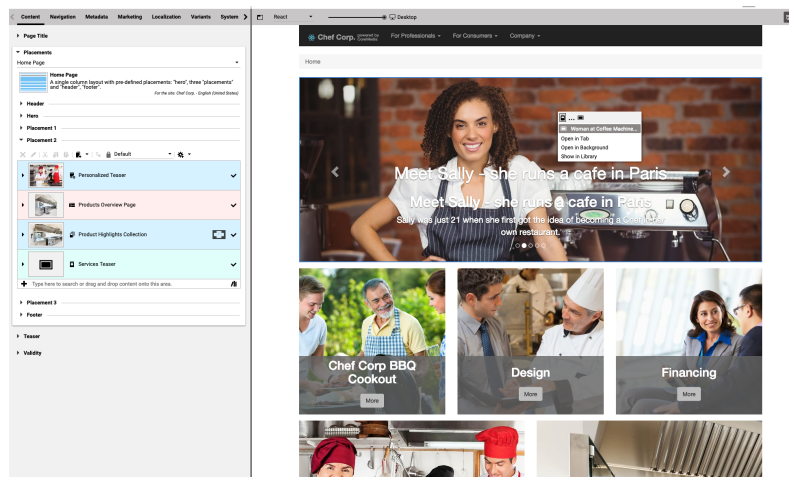


Figure 2.4. Integration of Single-Page-Applications (Headless Mode)

## 3. Overview of eCommerce Blueprint

The *eCommerce Blueprint* provides a modern, appealing, highly visual website template that can be used to start a customization project. It demonstrates the capability to build localizable, multi-national, experience-driven eCommerce web sites. Integration with HCL Commerce, SAP Hybris Commerce and Salesforce Commerce Cloud ships out of the box. Other eCommerce systems can be integrated via the CoreMedia eCommerce API as a project solution.

Based on a fully responsive, mobile-first design paradigm, the *eCommerce Blueprint* leverages most of our bricks and the FreeMarker templating framework. It scales from mobile via tablet to desktop viewport sizes and uses the CoreMedia Adaptive and Responsive Image Framework to dynamically deliver the right image sizes in the right aspect ratios and crops for each viewport.

The responsive navigation can blend commerce as well as content categories and content pages seamlessly and in any user-defined order that does not have to follow the catalog structure. Navigation nodes with URLs to external sites can be added in the content.

The following pictures show the standard integrations with Salesforce Commerce Cloud and SAP Hybris as well as with headless commerce systems.

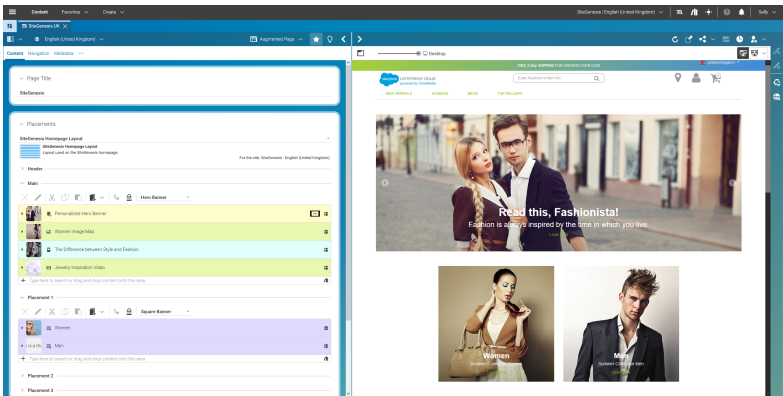


Figure 3.1. The integration of CoreMedia with Salesforce Commerce Cloud in CoreMedia Studio

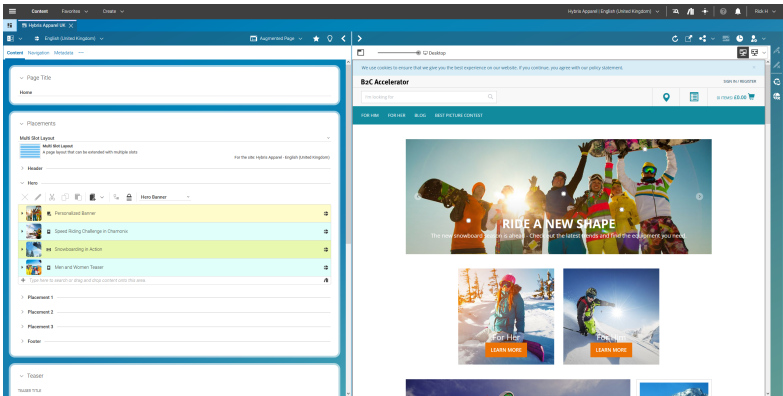
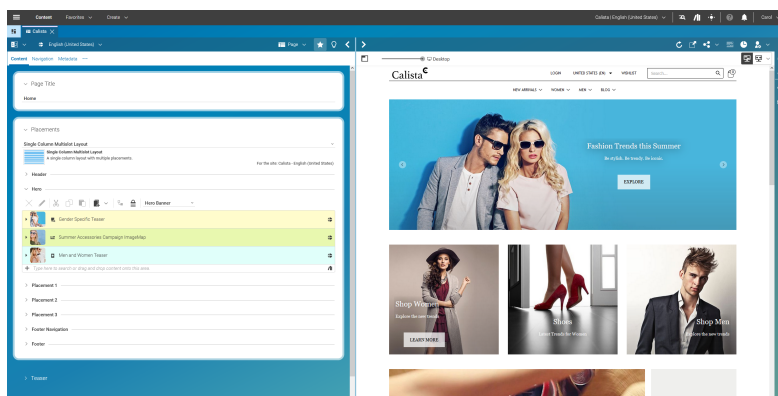


Figure 3.2. The integration of CoreMedia with SAP Hybris in CoreMedia Studio

## Overview of eCommerce Blueprint |



*Figure 3.3. The integration of CoreMedia with a headless commerce system (content-led integration scenario)*

## 3.1 CoreMedia Content Cloud for Salesforce Commerce Cloud

*CoreMedia Content Cloud* offers the commerce-led integration scenario with *Salesforce Commerce Cloud*. In the commerce-led scenario, pages are delivered by the *Salesforce Commerce Cloud* system.

The page navigation can be managed in CoreMedia – based on the catalog category, but allowing manual overrides, mixing content and commerce.

You can augment the categories and product detail pages with content from the CMS. Content and settings are also inherited along the catalog category structure.

### 3.1.1 Screenshots of the Integration – the Business User View

In the following, you will see a couple of screenshots showing the deep out-of-the-box integration with *Salesforce Commerce Cloud*.

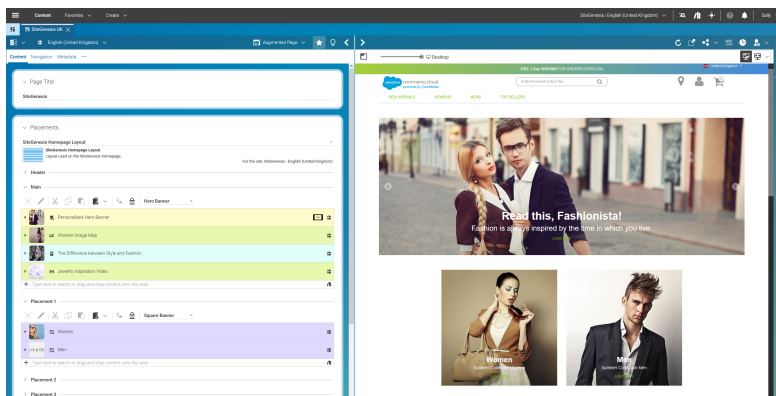


Figure 3.4. Using CoreMedia Studio to position inspirational content within Salesforce Commerce Cloud

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

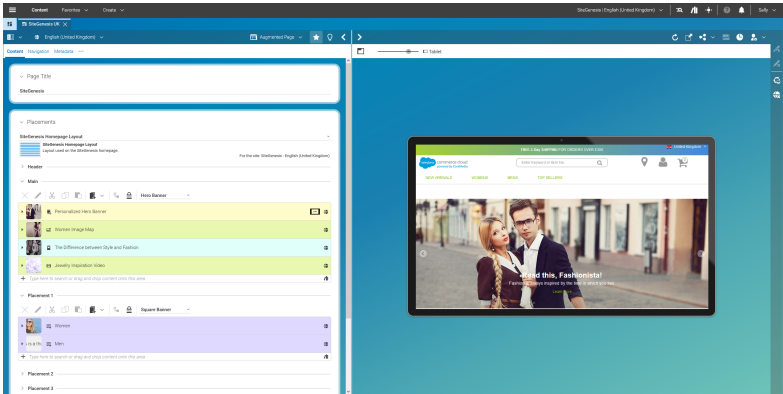


Figure 3.5. Responsive Preview of the storefront experience within CoreMedia Studio (here: tablet view)

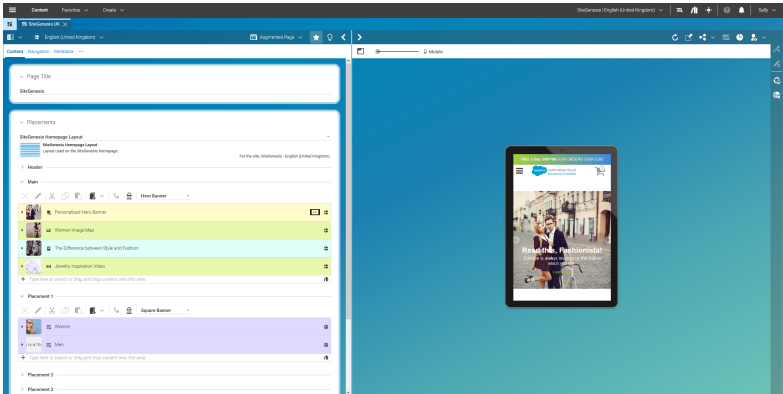


Figure 3.6. Responsive Preview of the storefront experience within CoreMedia Studio (here: mobile view)



Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

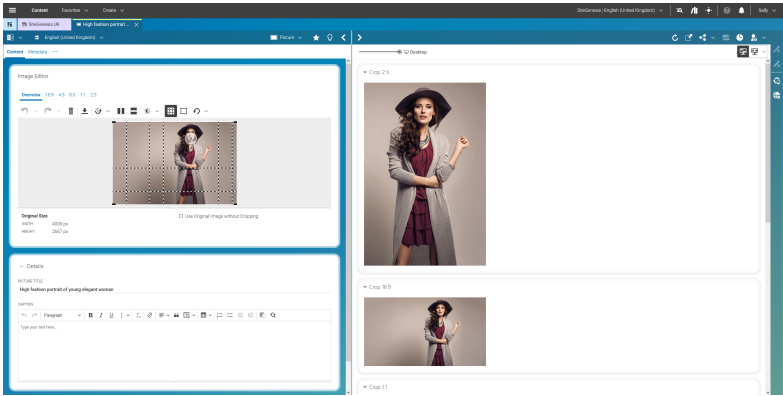


Figure 3.7. Image Cropping within CoreMedia Studio

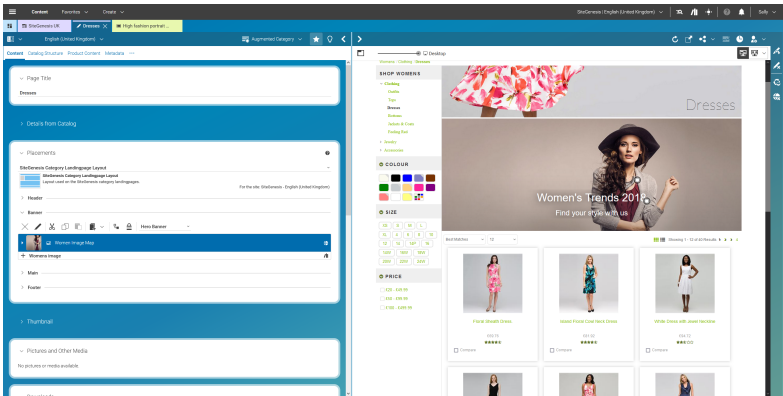


Figure 3.8. Adding inspirational content to Category/Listing Pages

# Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

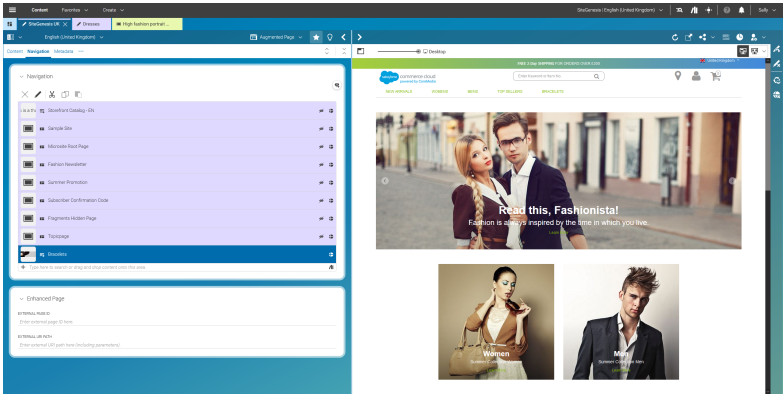


Figure 3.9. Managing the Navigation Menu within CoreMedia Studio

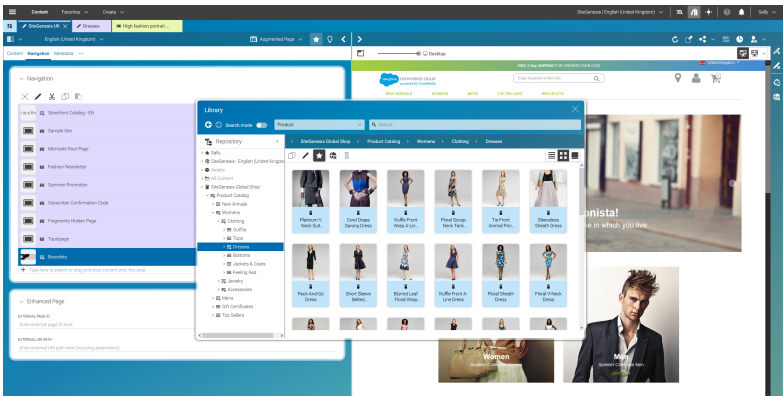


Figure 3.10. Product Catalog Access within CoreMedia Studio – incl. Product Variant Support

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

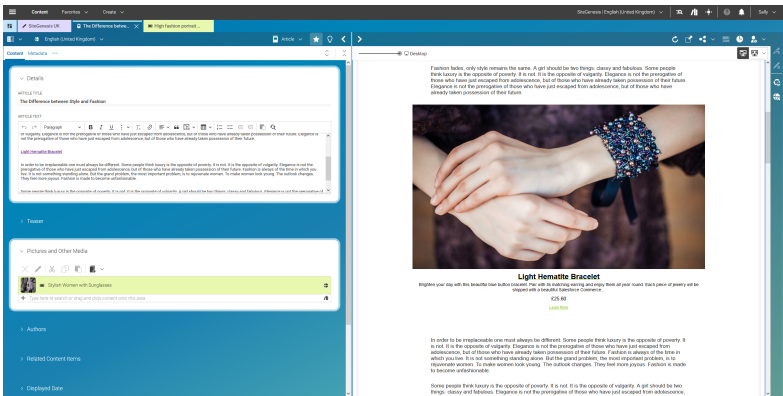


Figure 3.11. Embedding Product and Category Banners within Stories

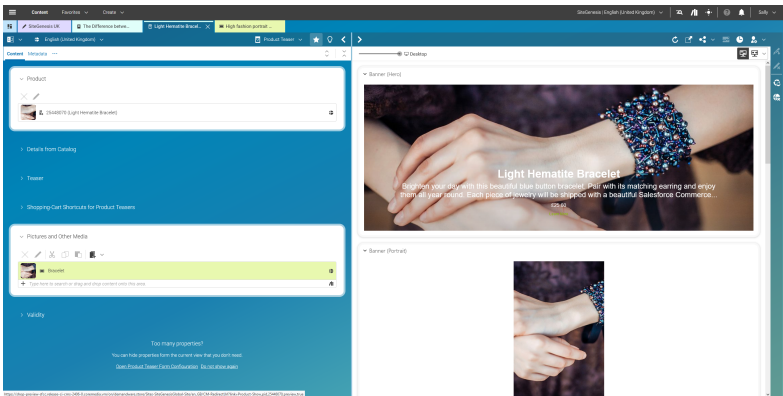


Figure 3.12. Enriching Standard Product Teasers

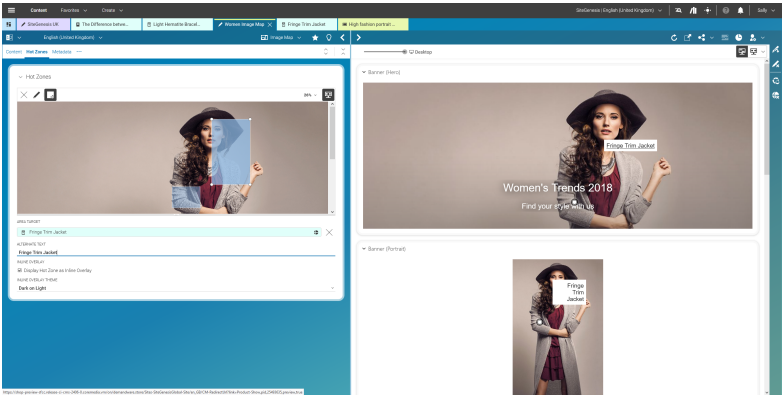


Figure 3.13. Shoppable Images – Creating Hot Zones within Images

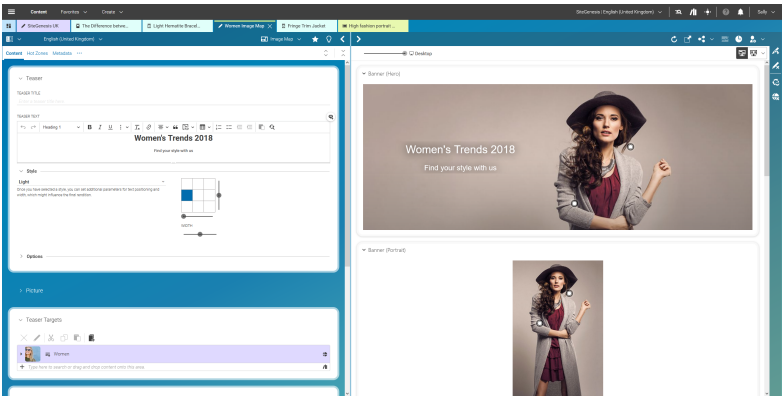


Figure 3.14. Advanced Text on Image Formatting and Positioning

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

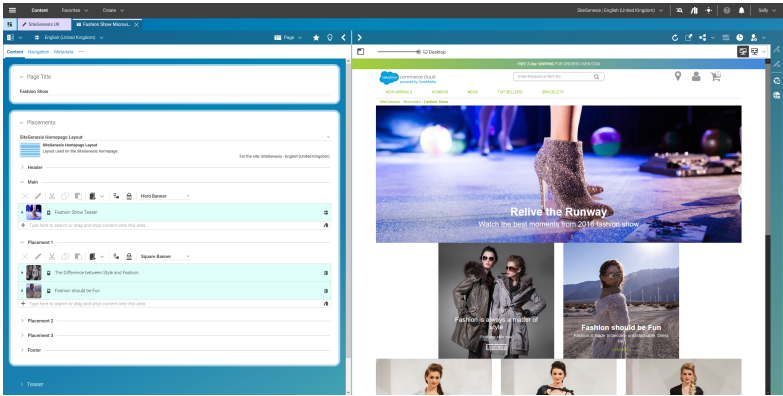


Figure 3.15. Creating content-driven landing pages within the Salesforce storefront

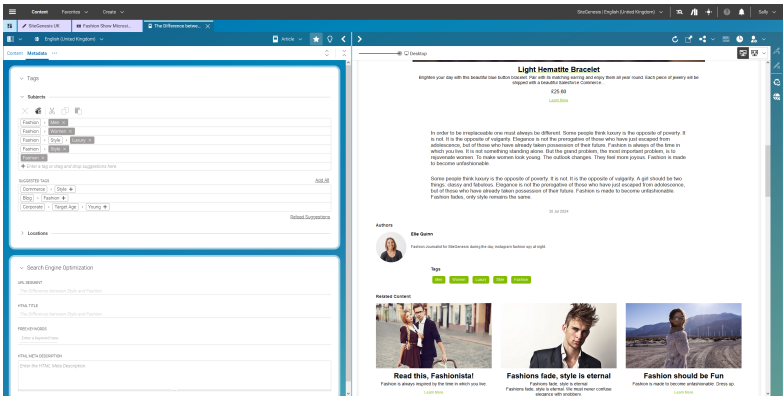


Figure 3.16. Advanced Tagging (incl. the option to connect to visual recognition services based on AI)

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

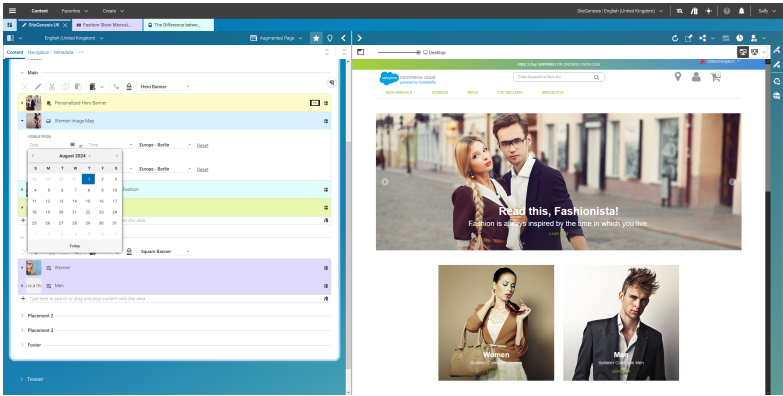


Figure 3.17. Scheduling Campaigns

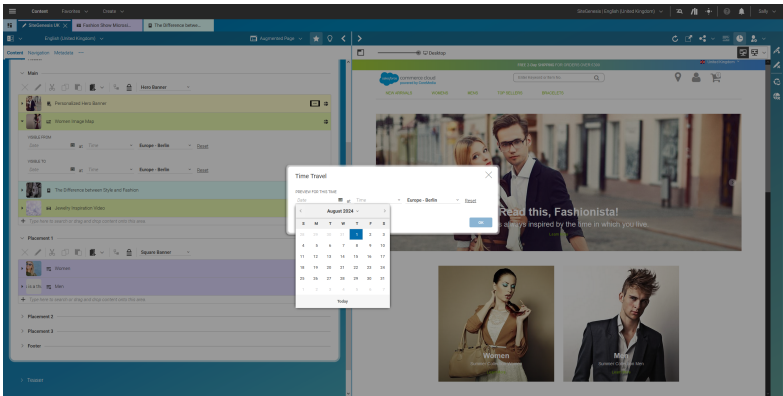


Figure 3.18. Time travel preview for scheduled campaigns

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

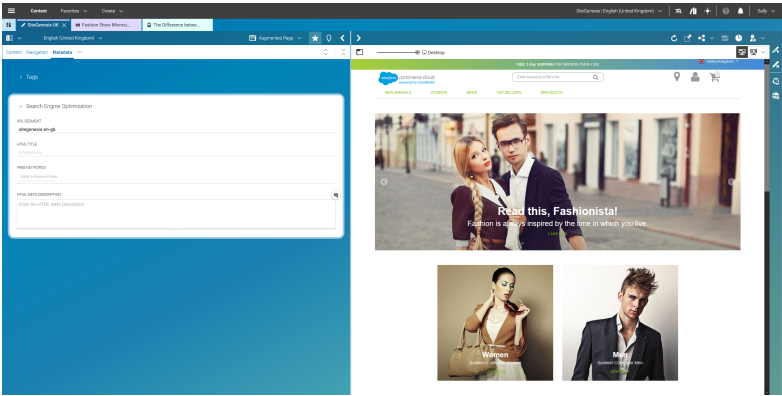


Figure 3.19. SEO – Human readable URLs and advanced metadata editing

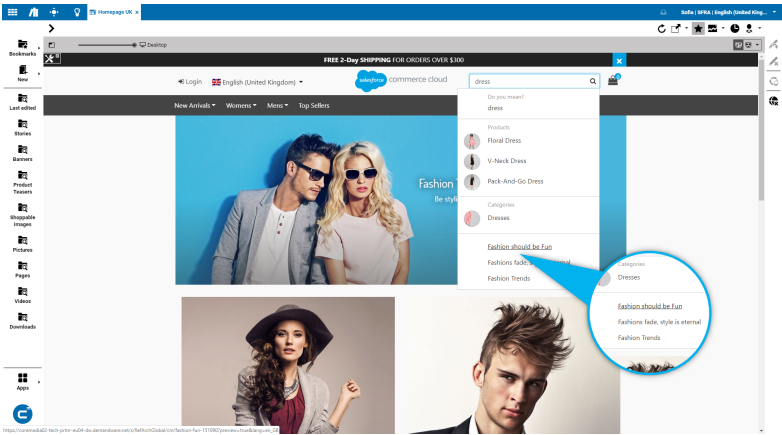


Figure 3.20. Non-product search

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

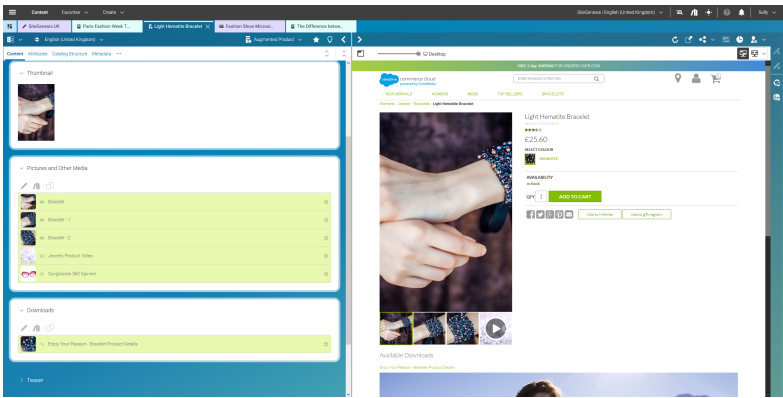


Figure 3.21. PDP Augmentation – Alternative Images, Videos, Downloads

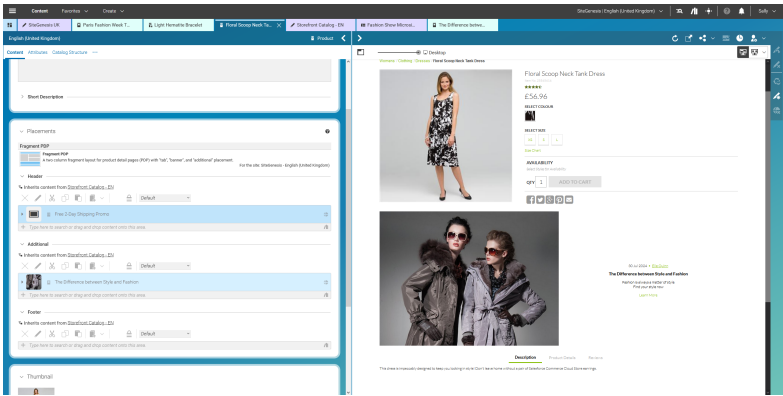


Figure 3.22. PDP Augmentation – Inspirational Content



Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

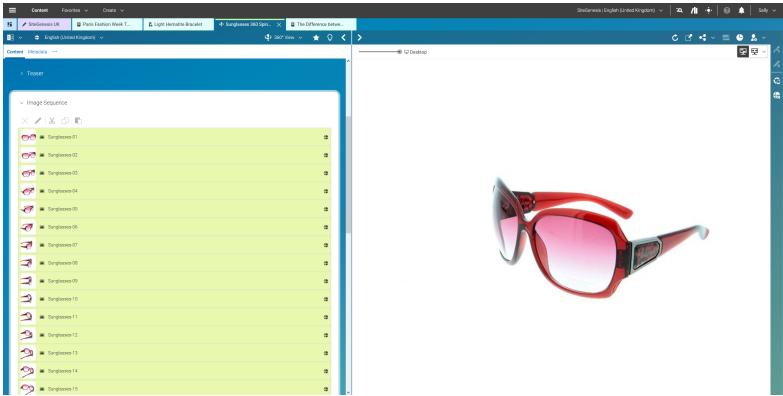


Figure 3.23. 360° Product Spinners

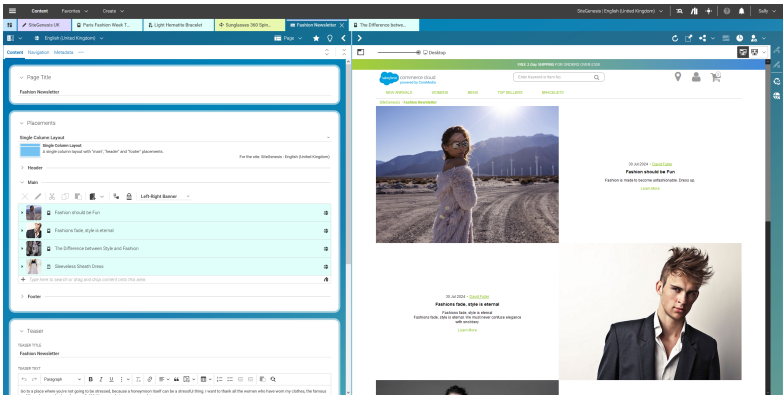


Figure 3.24. Content reuse for Newsletters / Outbound Marketing

# Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

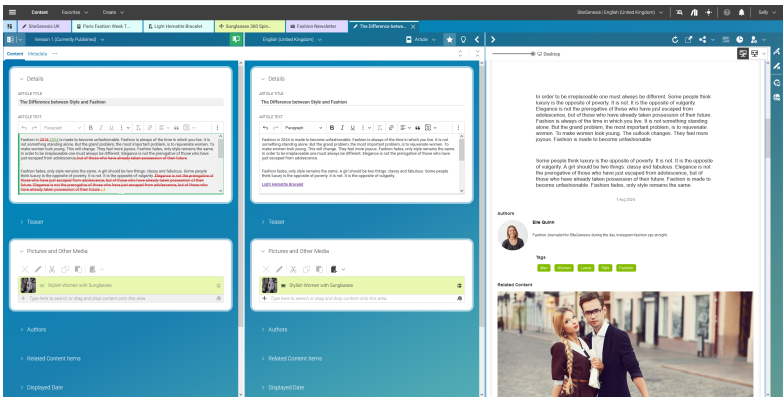


Figure 3.25. Content differencing

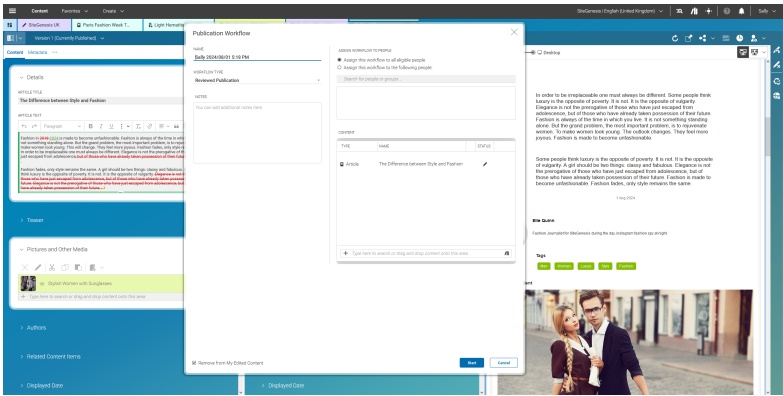


Figure 3.26. Reviewed Publication Workflows

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

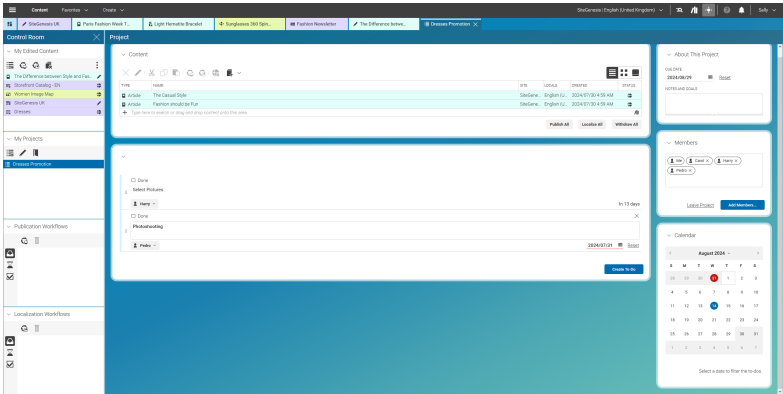


Figure 3.27. Collaboration in Teams

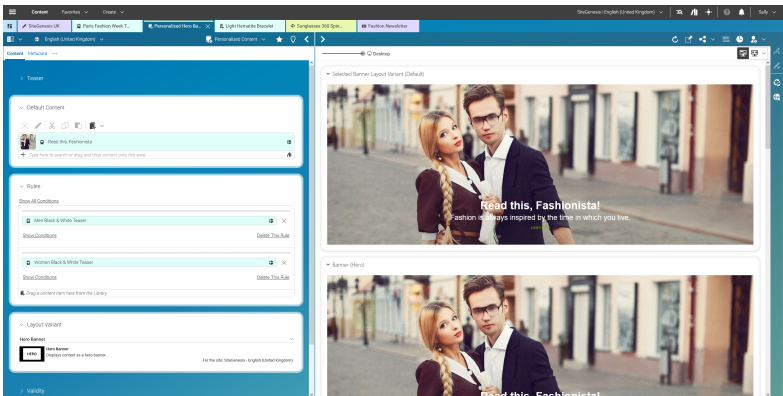


Figure 3.28. Personalization based on click behavior or customer groups within Salesforce

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

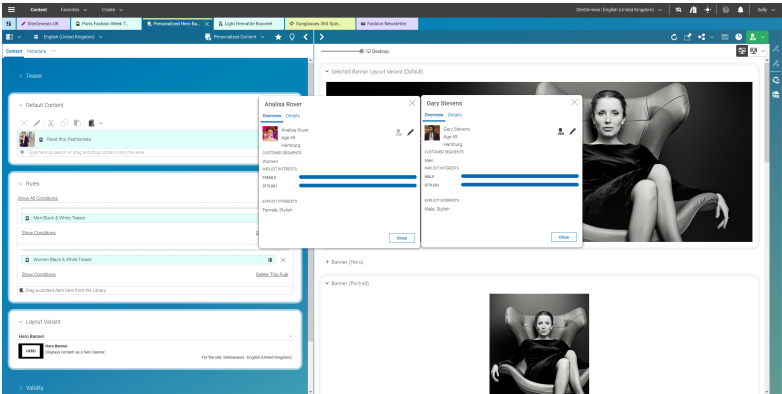


Figure 3.29. Persona-based Preview of the Experience

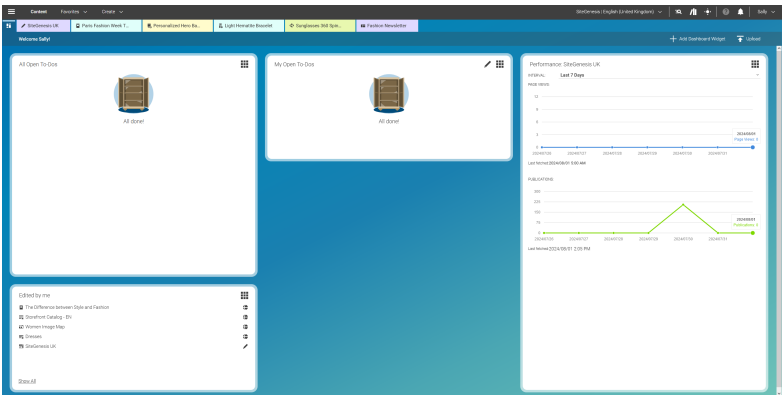


Figure 3.30. Dashboard Overview

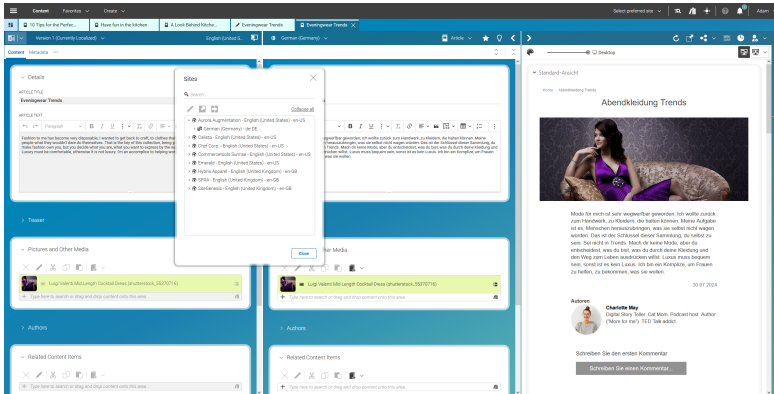


Figure 3.31. Strong Translation / Localization Capabilities

### 3.1.2 Adding CMS Fragments to Shop Pages

A pure eCommerce system is focused on the more transactional aspects of the buying process. To create a more engaging user experience you can augment the catalog pages with editorial content from the CMS. This includes, articles, images or videos.

There are two types of shop pages that can be extended by *CoreMedia Content Cloud*:

- **Catalog Pages** that are part of the catalog hierarchy, like a Category Overview or Landing Page and a Product Detail Page (PDP). They are extended by *Augmented Categories* and *Augmented Products* in the CMS.
- **Other Pages** that are not located in the catalog hierarchy. For example, all subordinate shop pages like "Contact Us", "Log On", "Checkout", "Register" or "Search Result", which also belong to a shop but don't have a category or a product connected with.

Even the homepage and other special topic pages belong to this type. These pages are extended by *Augmented Pages* in the CMS.

In addition, you can show complete CMS pages in the context of the commerce system. That page type is called **Content Pages**.

The basis for augmentation is the use of the *CoreMedia Content Widget* in content slots or the `islcinclude` tag in *ISML* templates.

*Types of augmentable pages*

*The augmentation process*

When you have prepared the shop-side with such content slots (either as *CoreMedia Content Widget* or directly with tags in shop templates), and the commerce system is properly connected with the CMS systems, you can now start augmenting shop pages in *Studio*.

### 3.1.3 CoreMedia Content Widget

The CoreMedia Content Widget is used to display content from the CoreMedia system on pages delivered by the eCommerce system. It is implemented as an extension of the *Salesforce* content slot mechanism. The slot configuration is extended with three custom attributes that can be filled when a content uses the CoreMedia Content Widget.

*Technical Background of the CoreMedia Content Widget*

Furthermore, there is an `ISML` template that must be executed when a content slot should be used for CoreMedia content.

#### Using the CoreMedia Content Widget

You can have one or more slots using a *CoreMedia Content Widget* per page. You might have, for example, a page with a main slot with content from the CMS or another page with a header and a footer coming from the CMS. The figure below shows a site from Salesforce SiteGenesis, that uses the *CoreMedia Content Widget*.

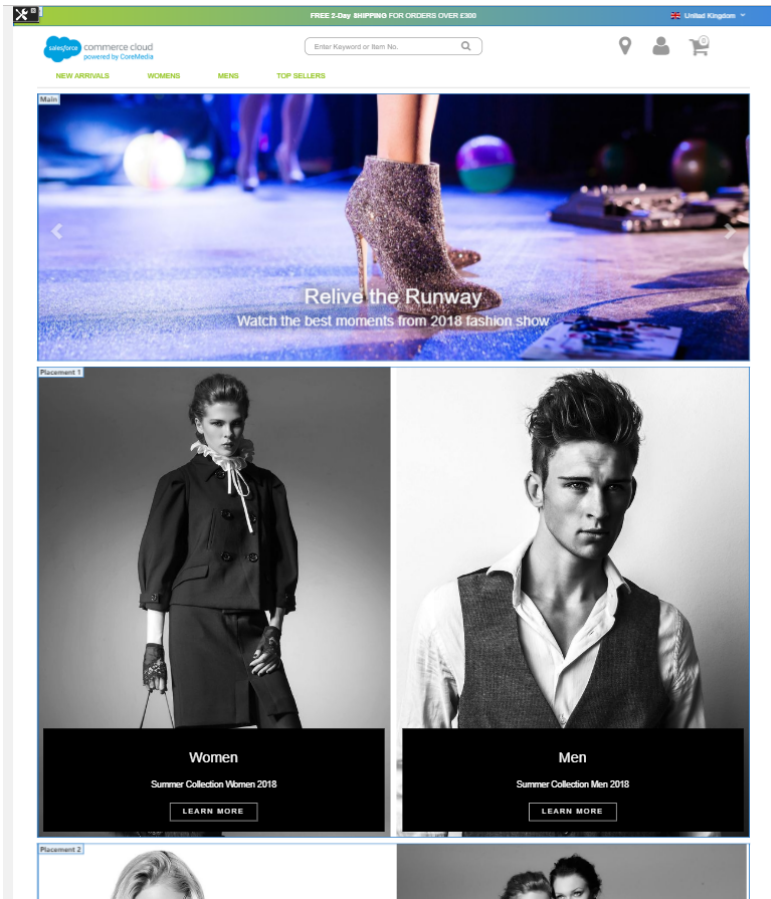


Figure 3.32. The integration of CoreMedia with a headless commerce system (content-led integration scenario)

## 3.1.4 Studio Integration of Commerce Content

In *CoreMedia Content Cloud* each content site can be configured with a specific shop instance to deliver content pages mixed with Commerce catalog items. The term "Commerce catalog items" means all items that live only in the com-

merce catalog. Nevertheless, these elements are to be interwoven with content on mixed pages.

From classical shop pages, like a product catalog ordered by categories or product detail pages up to landing pages or homepages, all grades of mixing content with catalog items are conceivable. The approach followed in this chapter, assumes that items from the catalog will be linked or embedded without having stored these items in the CMS system. Catalog items will be linked typically and not imported.

### 3.1.4.1 Catalog View in CoreMedia Studio Library

## Catalog View in CoreMedia Studio Library

When the connection to a *Salesforce Commerce* system and a concrete shop for a content site are configured, the *Studio* Library shows the commerce catalog to browse product categories and products in the commerce catalog and to search for products and product variants. After the editor has selected a preferred site with a valid store configuration the catalog view will be enabled and the catalog will be shown in the Library:

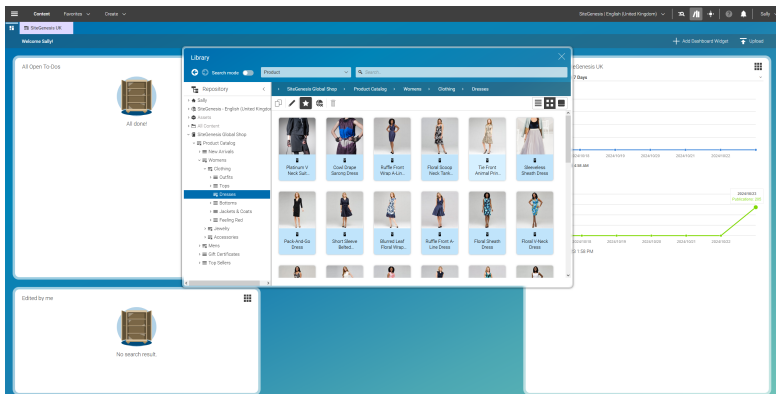


Figure 3.33. Library with catalog in the tree view

In some catalogs it is possible to put a category on multiple places within the catalog tree. But the Commerce Hub ensures that a category can only have one home (a unique parent category). All additional occurrences of a category are shown as a link in the tree. If you click on such a link node you will automatically end up at the place in the tree where the category is actually at home.



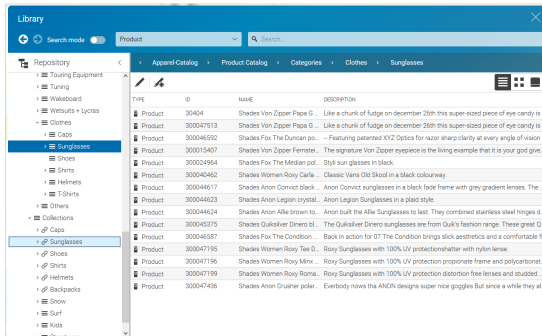


Figure 3.34. Library tree with multiple occurrences of the same category

These catalog items can be accessed and assigned to various places within your content. For example, an *eCommerce Product Teaser* content item can link to a product or product variant from the catalog. The product link field (in *eCommerce Product Teaser* content item) can be filled by drag and drop from the library in catalog mode.

Linking a content (like the *eCommerce Product Teaser*) to a catalog item leads to a link that is stored in the CMS content item and references the external element. Apart from the external reference (in the case of the commerce system it is typically a persistent identifier like the product code for products) no further data will be imported (importless integration).

While browsing through the catalog tree you can also open a preview of a category or a product from the library. Simply double-click on a product in the product list or use the context menu on a product or a category and choose the entry **Open in Tab** from the context menu as shown in the pictures below.

In addition to the ability to browse through the commerce catalog in an explorer-like view it is also possible to search for products and variants from catalog. As for the content search if you are in the catalog mode and you type a search keyword into the search field and press **Enter**, the search in the commerce system will be triggered and a search result displayed.

## 3.1.4.2 Commerce related Preview Support Features

*CoreMedia Studio* supports a variety of commerce preview functions directly:

- Time based preview (time travel)

When a preview date is set in *CoreMedia Studio*, it sets the virtual render time to a time in the future. If the currently previewed page contains content from *Hybris Commerce*, it is desirable that also these content reflects the given preview time. That could be a certain validity period of a product or another display rule that influences the displayed catalog items.

- Customer segment based preview

The feature segment based preview supports the creation of personalized content. In this case, content is shown depending on the membership in specific customer segments. In addition to the existing rules, you can define rules that are based on the belonging to customer segments that are maintained by the commerce system.

These commerce segments will be automatically integrated and appear in the chooser if you create a new rule in a personalized content. For a preview, editors can use test personas which are associated with specific customer segments.

Figure 3.35, “Test Customer Persona with Commerce Customer Segments” [53] shows an example where the test persona is female and has already been registered.

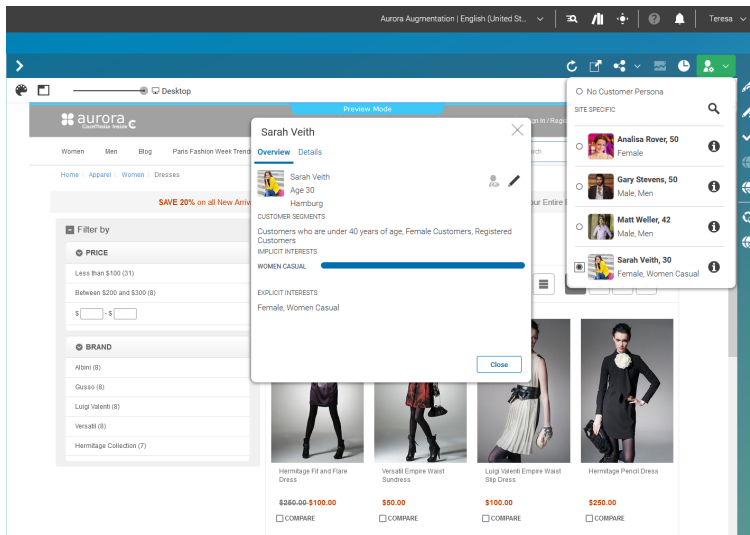


Figure 3.35. Test Customer Persona with Commerce Customer Segments

Such preview settings apply as long as they are not reset by the editor.

The test persona content can be created and edited in *CoreMedia Studio*. The customer segments available for selection will be automatically read from the commerce system. By default, all user segments available in the eCommerce system are displayed for selection. Under some circumstances it may be desirable to restrict the shown user segments, for instance for studio performance reasons or for better clarity for the editor. See ????

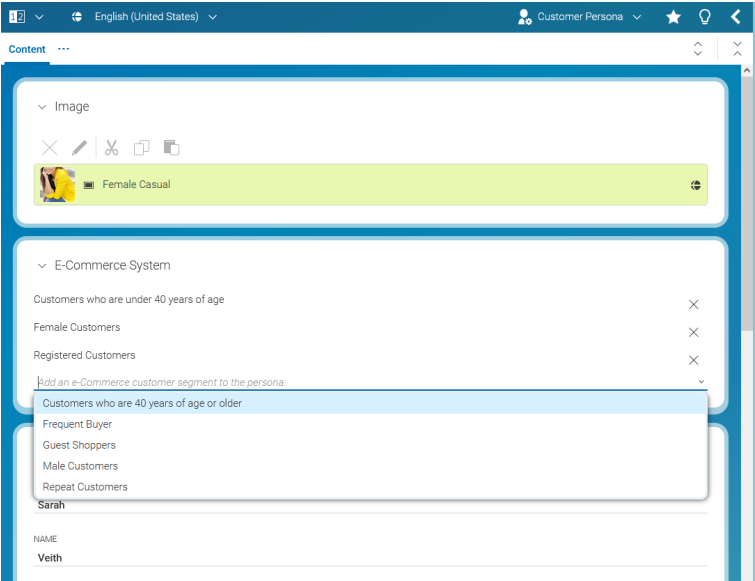


Figure 3.36. Edit Commerce Segments in Test Customer Persona

The commerce segments that the current user belongs to are available during the rendering process within a *CoreMedia CAE*. Thus, content from the CoreMedia system can also be filtered based on the current commerce segments.

In the other direction, if the personalized content is integrated within a content fragment on a shop page, the current commerce user is also transmitted as a parameter. Thus, the CoreMedia system can retrieve the connected customer segments from the commerce system in order to perform commerce segment personalization within the supplied content fragments.

### 3.1.4.3 Selecting a Layout for an Augmented Page

*CoreMedia Content Cloud* comes with a predefined set of page layouts. Typically, this selection will be adapted to your needs in a project. By selecting a layout an editor specifies which placements the new page will have, which of them can be edited and how the placements are arranged generally. It should correspond to the actual shop page layout. All usable placements should be addressed. The placement names must match the placement names used in the slot definition on the shop side.

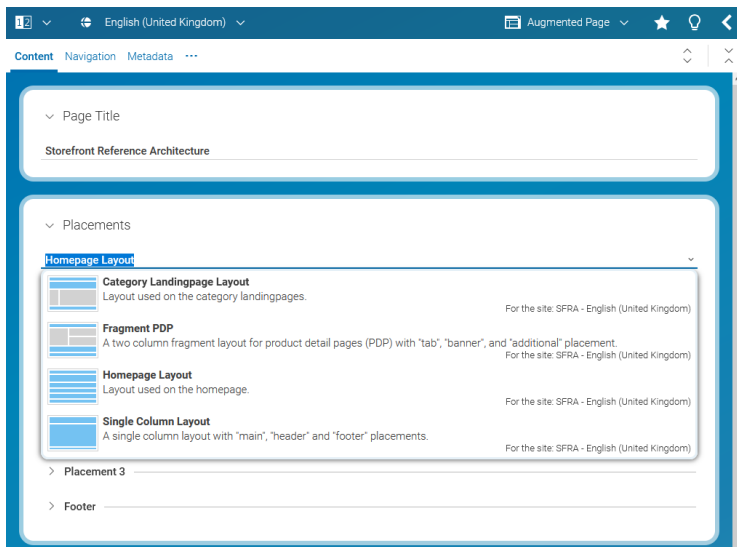


Figure 3.37. Choosing a page layout for a shop page

If you augment a category, the corresponding *Augmented Category* content item contains two page layouts: the one in the *Content* tab is applied to the *Category Overview Page* and the other in the *Product Content* tab is used for all *Product Detail Pages*. Both layouts are taken from the root category. The layouts that are set there form the default layouts for a site. Hence, they should be the most commonly used layouts. If you want something different, you can choose another layout from the list.

## 3.2 CoreMedia Content Cloud for SAP Hybris

CoreMedia Content Cloud for SAP Hybris offers the commerce-led integration scenario with SAP Hybris Commerce.

In the commerce-led scenario, pages are delivered by the SAP Hybris Commerce system. You can augment the homepage, category pages and product detail pages with content from the CMS.

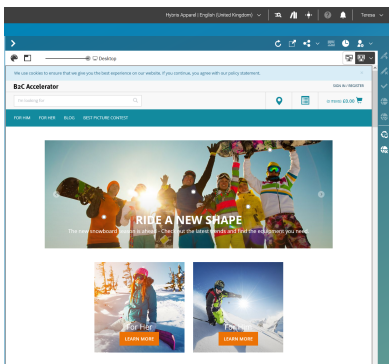


Figure 3.38. SAP Hybris example

### 3.2.1 Screenshots of the Integration – the Business User View

In the following, you will see a couple of screenshots showing the deep out-of-the-box integration with SAP Hybris.

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

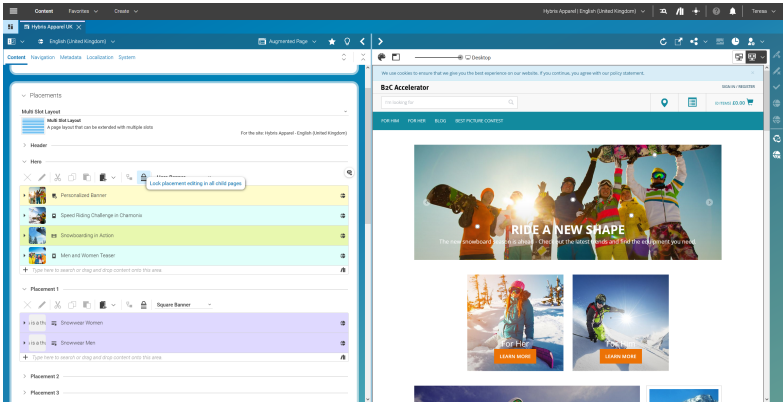


Figure 3.39. Using CoreMedia Studio to position inspirational content within SAP Hybris

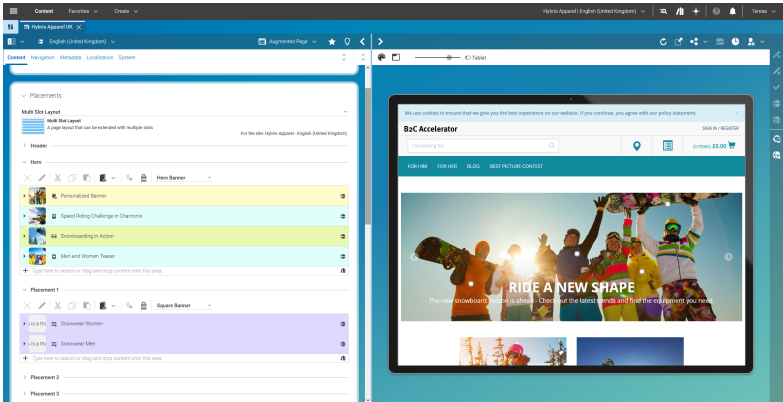


Figure 3.40. Responsive Preview of the storefront experience within CoreMedia Studio (here: tablet view)

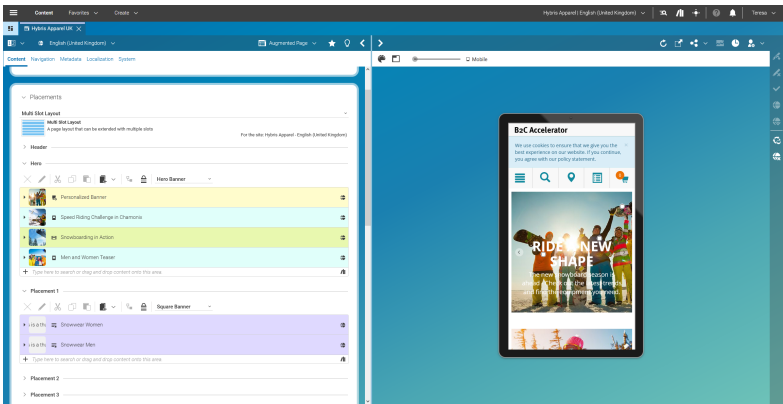


Figure 3.41. Responsive Preview of the storefront experience within CoreMedia Studio (here: mobile view)

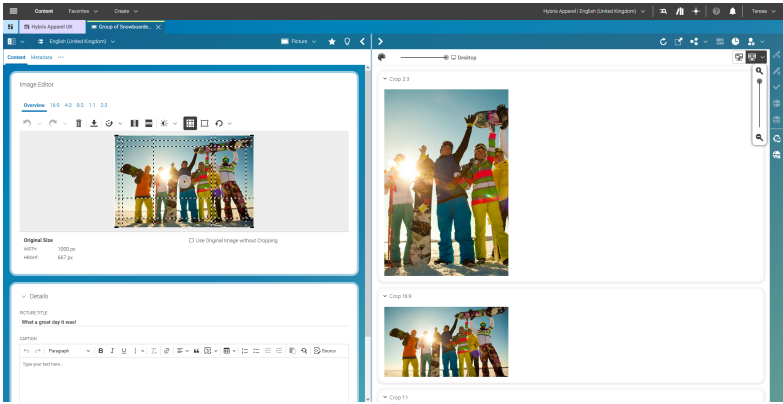


Figure 3.42. Image Cropping within CoreMedia Studio

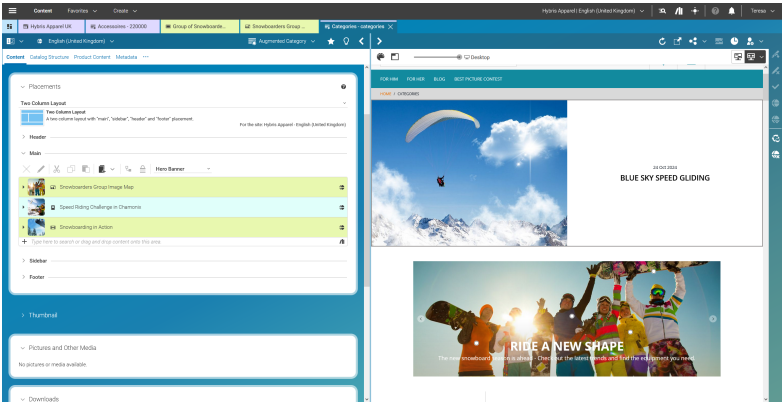


Figure 3.43. Adding inspirational content to Category/Listing Pages

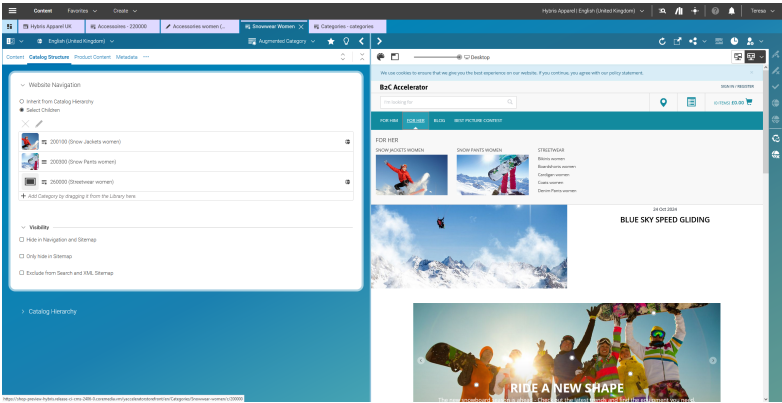


Figure 3.44. Managing the Navigation Menu within CoreMedia Studio



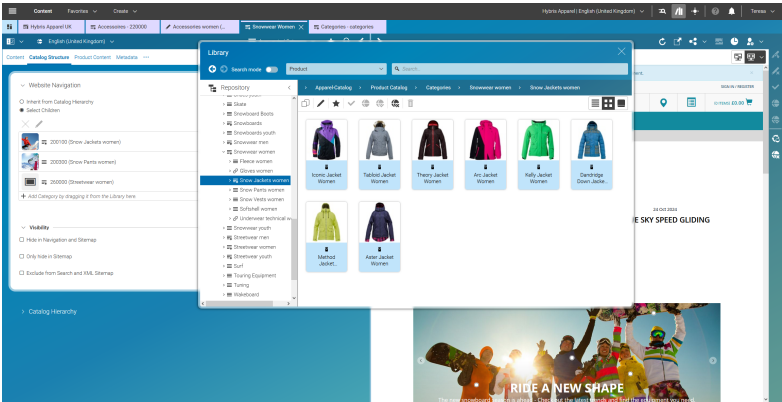


Figure 3.45. Product Catalog Access within CoreMedia Studio – incl. Product Variant Support

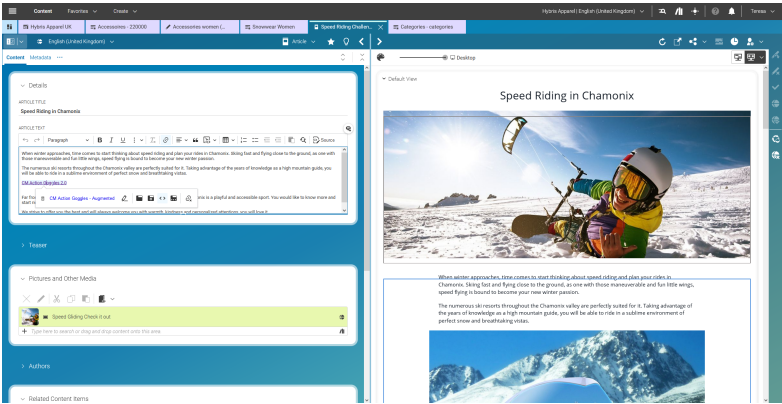


Figure 3.46. Embedding Product and Category Banners within Stories

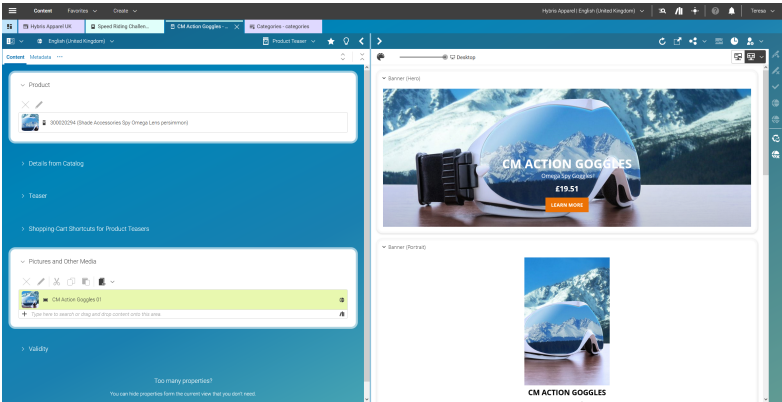


Figure 3.47. Enriching Standard Product Teasers

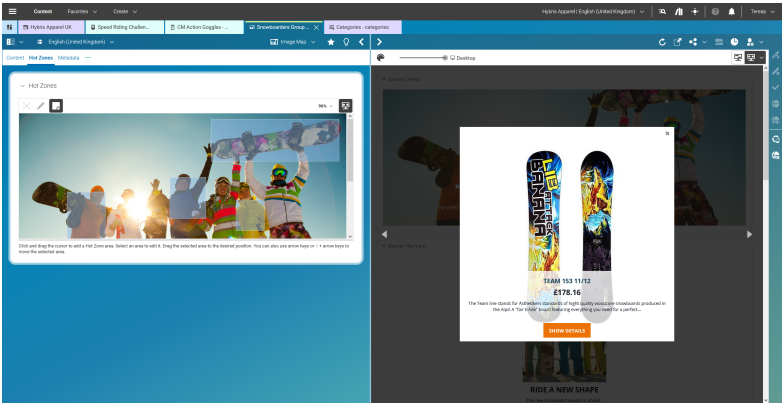


Figure 3.48. Shoppable Images – Creating Hot Zones within Images

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

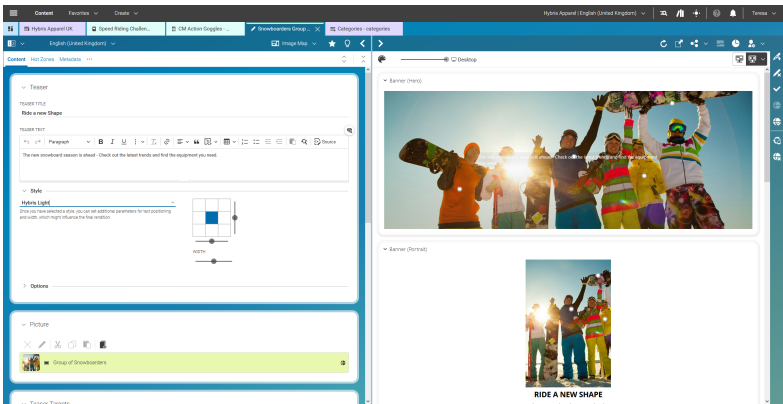


Figure 3.49. Advanced Text on Image Formatting and Positioning

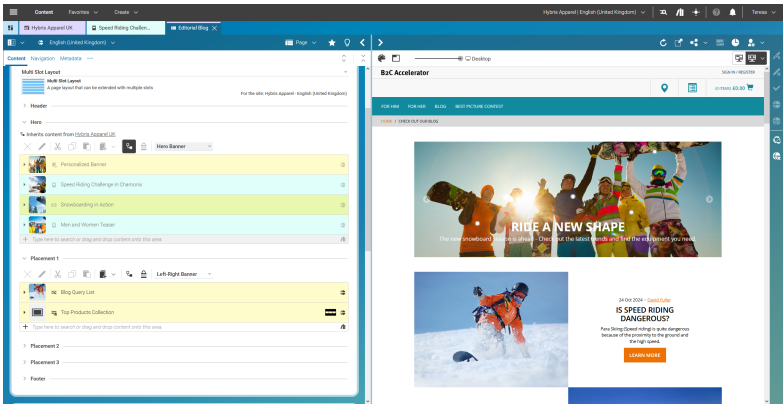


Figure 3.50. Creating content-driven landing pages

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

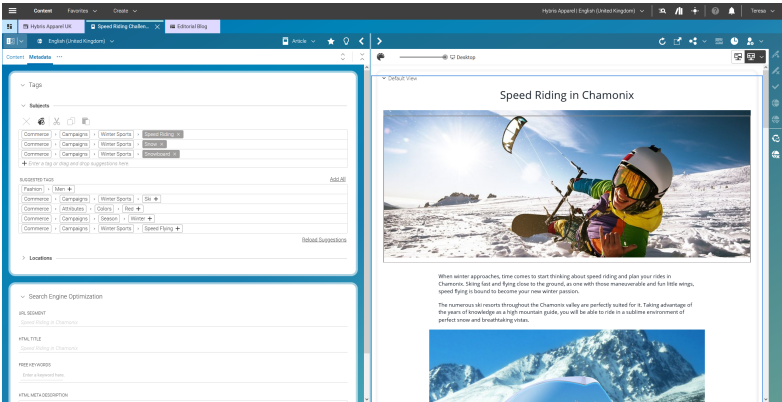


Figure 3.51. Advanced Tagging (incl. the option to connect to visual recognition services based on AI)

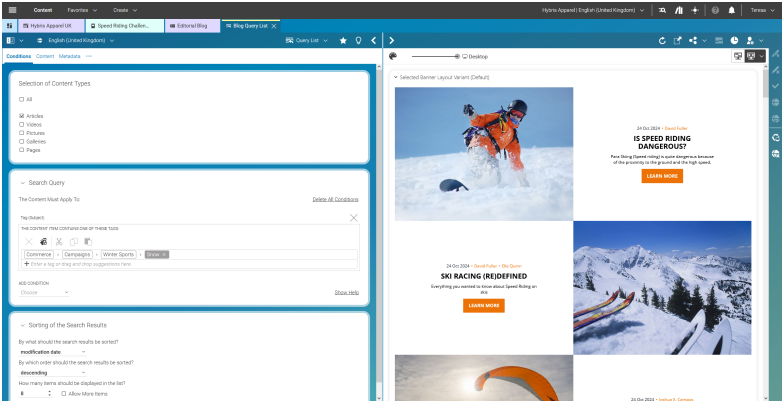


Figure 3.52. Rule-Driven Content Lists – Dynamic Lists

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

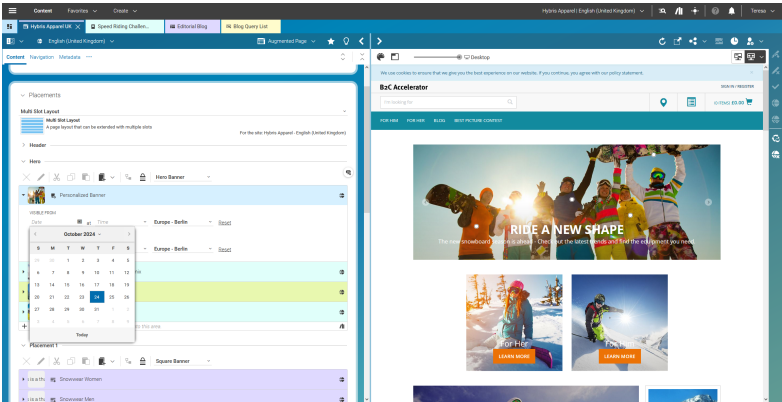


Figure 3.53. Scheduling Campaigns

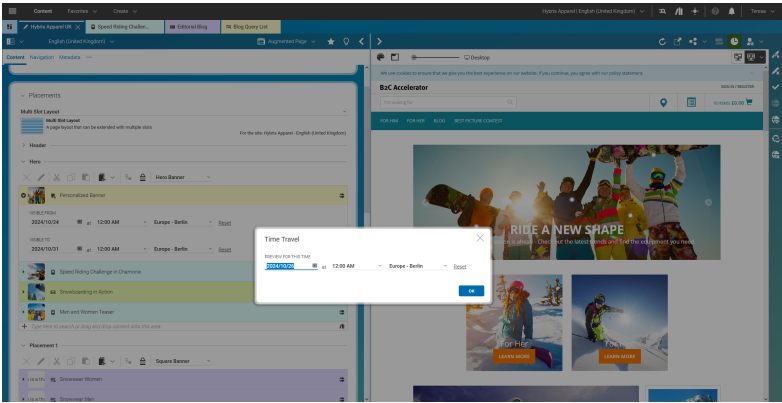


Figure 3.54. Time travel preview for scheduled campaigns

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

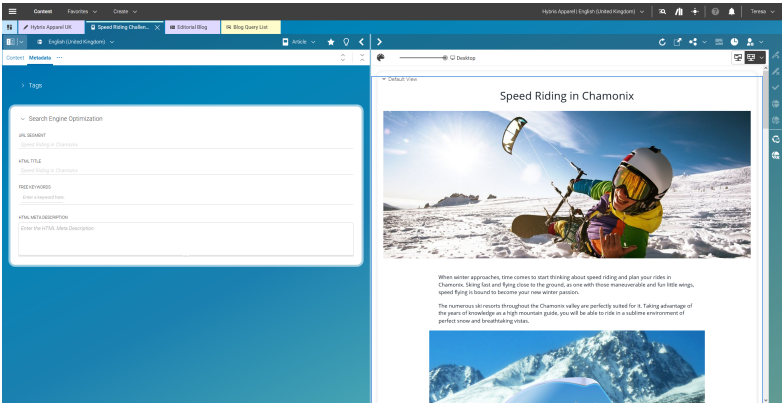


Figure 3.55. SEO – Human readable URLs and advanced metadata editing

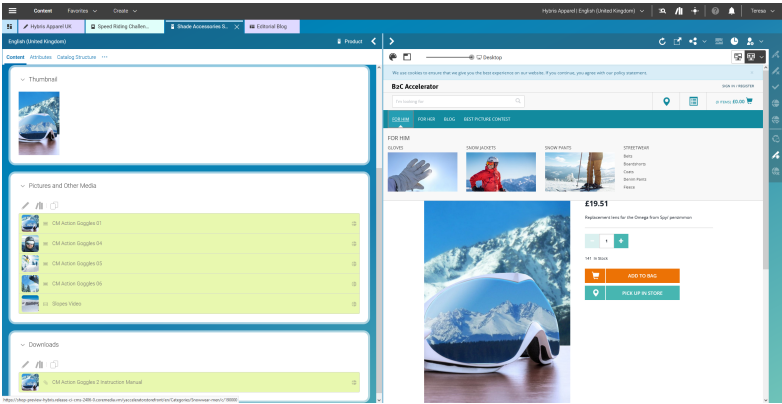


Figure 3.56. PDP Augmentation – Alternative Images, Videos, Downloads

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

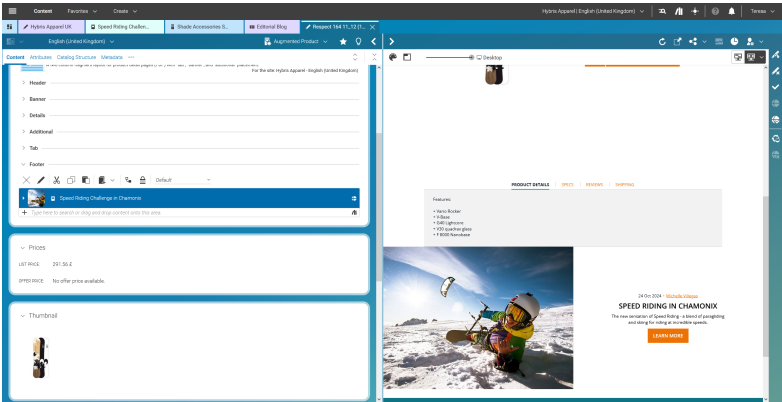


Figure 3.57. PDP Augmentation – Inspirational Content

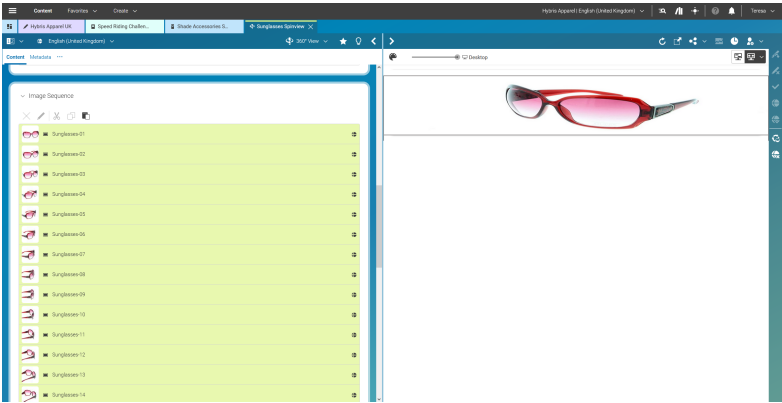


Figure 3.58. 360° Product Spinners

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

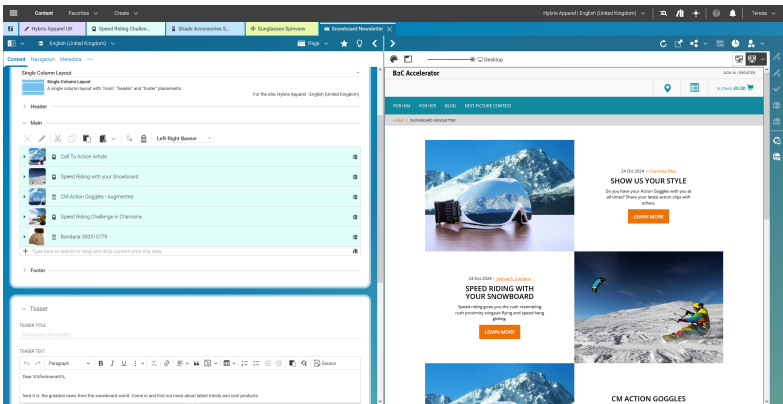


Figure 3.59. Content reuse for Newsletters / Outbound Marketing

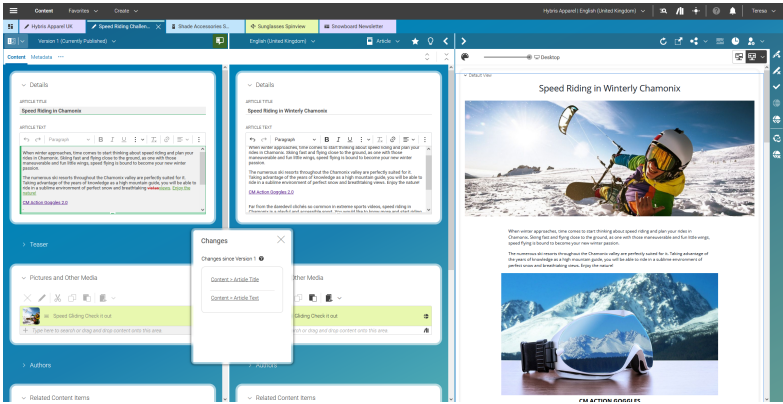


Figure 3.60. Content differencing



Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

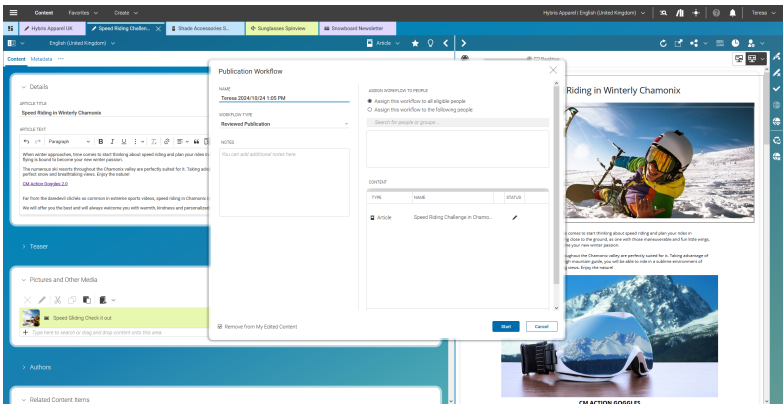


Figure 3.61. Reviewed Publication Workflows

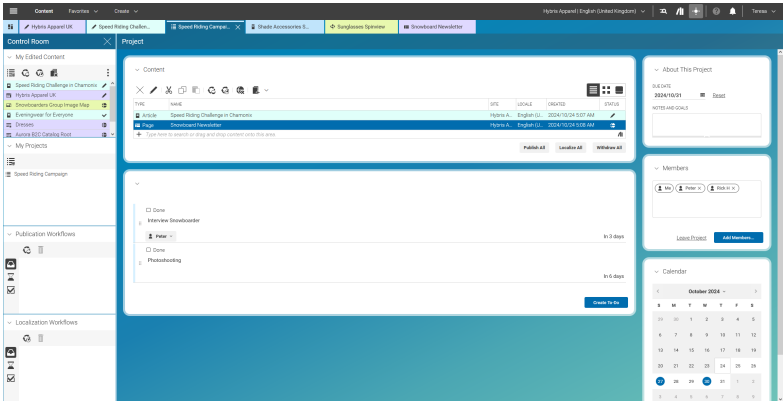


Figure 3.62. Collaboration in Teams

Overview of eCommerce Blueprint | Screenshots of the Integration – the Business User View

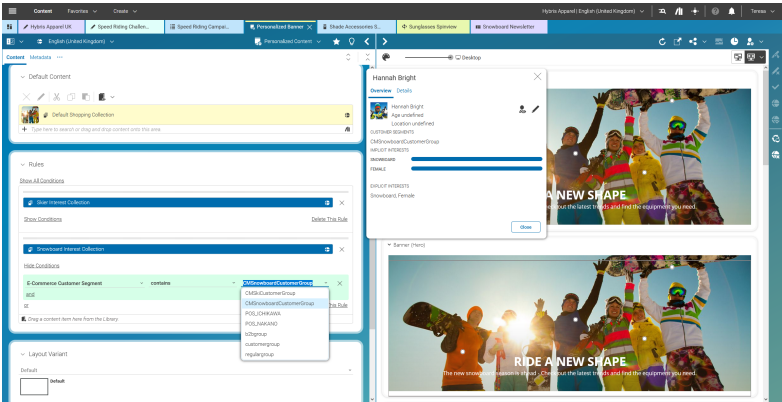


Figure 3.63. Personalization based on click behaviour or customer segments within SAP Hybris

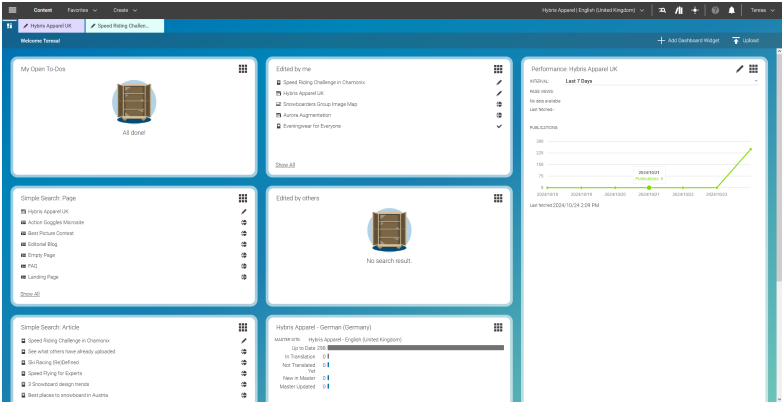


Figure 3.64. Dashboard Overview

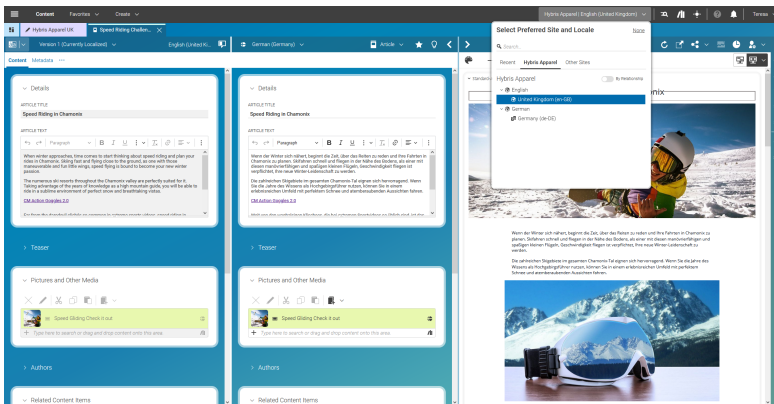


Figure 3.65. Strong Translation / Localization Capabilities

### 3.2.2 Adding CMS Fragments to Shop Pages

A pure eCommerce system is focused on the more transactional aspects of the buying process. To create a more engaging user experience you can augment the catalog pages with editorial content from the CMS. This includes, articles, images or videos.

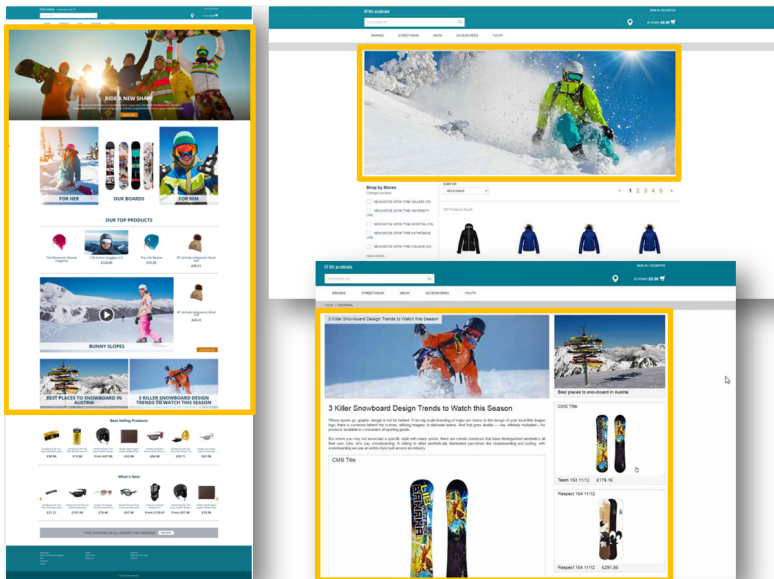


Figure 3.66. Various Shop Pages with CMS Fragments

There are two types of shop pages that can be extended by the CoreMedia CMS:

- **Catalog Pages** that are part of the catalog hierarchy, like a Category Overview or Landing Page and a Product Detail Page (PDP). They are extended by **Augmented Categories** and **Augmented Products** in the CMS.
- **Other Pages** that are not located in the catalog hierarchy. For example, all subordinate shop pages like "Contact Us", "Log On", "Checkout", "Register" or "Search Result", which also belong to a shop but don't have a category or an product connected with. Even the homepage and other special topic pages belong to this type. These pages are extended by **Augmented Pages** in the CMS.

In addition, you can show complete CMS pages in the context of the commerce system. That page type is called **Content Pages**.

The basis for augmentation is the use of the *CoreMedia Content Widget* or the `lc:include` tag in the commerce system.

*The augmentation process*

On the commerce side, add the *CoreMedia Content Widget* to the commerce page layouts or write the `lc:include` tag directly into a shop template. The value of the placement property corresponds to the placement name within a CMS-side page layout. Technically, the *CoreMedia Content Widget* uses also the `lc:include` tag internally.

When you have prepared the shop-side with such content slots (either as *CoreMedia Content Widget* or directly with tags in shop templates), and the commerce system is properly connected with the CMS systems, you can now start augmenting shop pages in *Studio*.

### 3.2.3 CoreMedia Content Widget

On the *Hybris Commerce* side it is necessary to define slots where the CMS content can be displayed. This is normally done by adding the *CoreMedia Content Widget* to a *Hybris Commerce* page layout. The tool with which this can be done is the *SAP SmartEdit*.

*Adding the CoreMedia Content Widget*

Take the Apparel-UK homepage page as an example. As you can see in the screenshot below, there is one *CoreMedia Content Widget* placed.

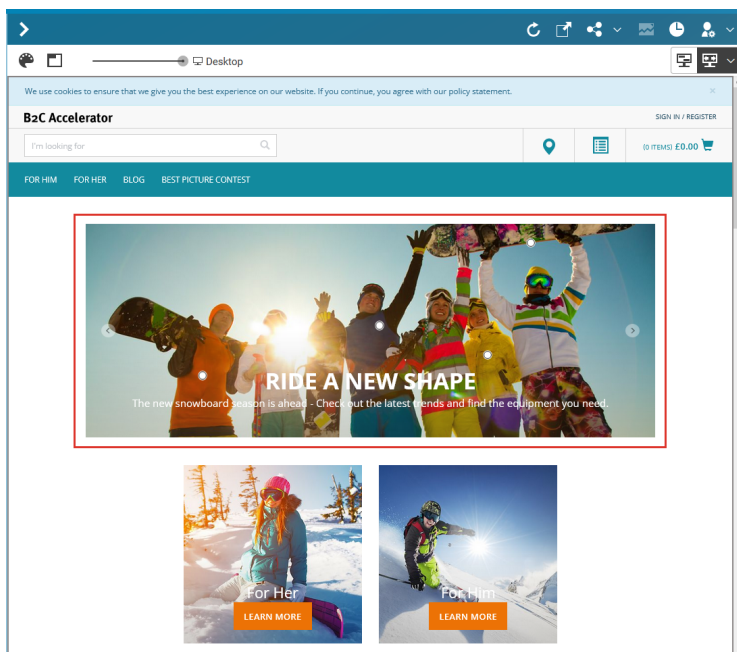


Figure 3.67. Using the CoreMedia Content Widget – A Homepage Fragment

The content that is shown in the *CoreMedia Content Widget* is taken from a placement of an augmenting CMS page. The name of the placement in the CMS page needs to correspond to the name configured in the *CoreMedia Content Widget*.

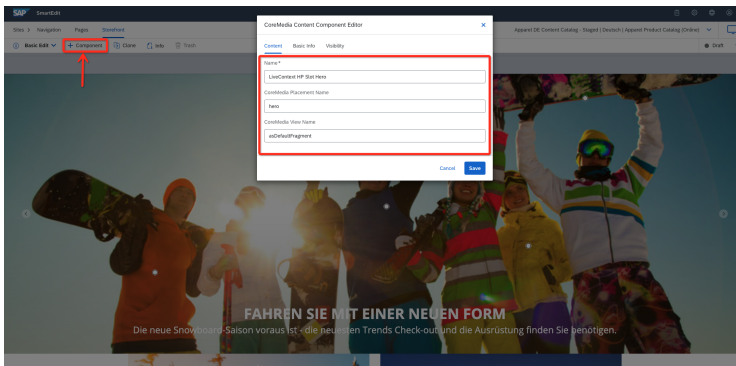


Figure 3.68. Using the CoreMedia Content Widget – Connection to CMS Content via placement name

## 3.2.4 Studio Integration of Commerce Content

*CoreMedia Content Cloud* integrates with SAP Hybris Commerce. In the following it is simply called the "commerce system" or "the shop".

Each content site can be configured with a specific shop instance to deliver content pages mixed with Commerce catalog items. The term "Commerce catalog items" means all items that live only in the commerce catalog. Nevertheless, these elements are to be interwoven with content on mixed pages.

From classical shop pages, like a product catalog ordered by categories or product detail pages up to landing pages or homepages, all grades of mixing content with catalog items are conceivable. The approach followed in this chapter, assumes that items from the catalog will be linked or embedded without having stored these items in the CMS system. Catalog items will be linked typically and not imported (importless integration).

### Catalog View in CoreMedia Studio Library

When the connection to a *Hybris Commerce* system and a concrete shop for a content site are configured the *Studio Library* shows the commerce catalog to browse product categories and products in the commerce catalog and to search for products and product variants. After the editor has selected a preferred site with a valid store configuration the catalog view will be enabled and the catalog will be shown in the Library:

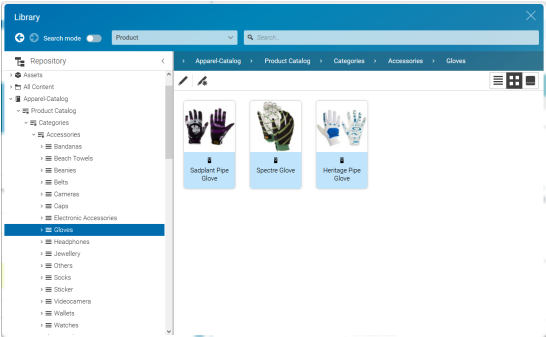


Figure 3.69. Library with catalog in the tree view

In some catalogs it is possible to put a category on multiple places within the catalog tree. But the Commerce Hub ensures that a category can only have one home (a unique parent category). All additional occurrences of a category are shown as a link in the tree. If you click on such a link node you will automatically end up at the place in the tree where the category is actually at home.

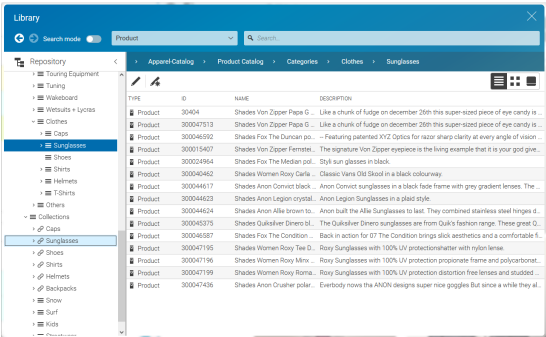


Figure 3.70. Library tree with multiple occurrences of the same category

These catalog items can be accessed and assigned to various places within your content. For example, an *eCommerce Product Teaser* content item can link to a product or product variant from the catalog. The product link field (in *eCommerce Product Teaser* content items) can be filled by drag and drop from the library in catalog mode.

Linking a content (like the *eCommerce Product Teaser*) to a catalog item leads to a link that is stored in the CMS content item and references the external element. Apart from the external reference (in the case of the commerce system it is typically a persistent identifier like the product code for products) no further data will be imported (importless integration).

While browsing through the catalog tree you can also open a preview of a category or a product from the library. Simply double-click on a product in the product list or use the context menu on a product or a category and choose the entry **Open in Tab** from the context menu as shown in the pictures below.

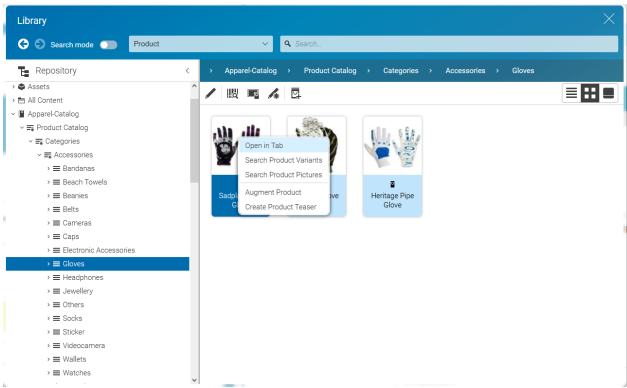


Figure 3.71. Open Product in tab

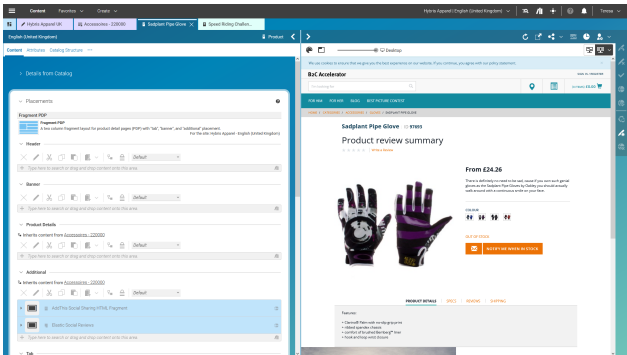


Figure 3.72. Product in tab preview



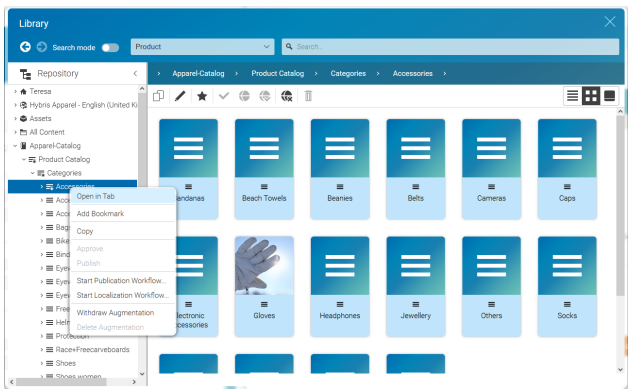


Figure 3.73. Open Category in tab

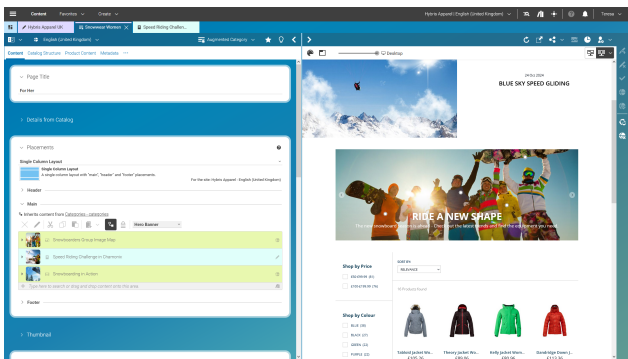


Figure 3.74. Category in tab preview

In addition to the ability to browse through the commerce catalog in an explorer-like view it is also possible to search for products and variants from catalog. As for the content search if you are in the catalog mode and you type a search keyword into the search field and press **Enter**, the search in the commerce system will be triggered and a search result displayed.

## Commerce related Preview Support Features

CoreMedia Studio supports a variety of commerce preview functions directly:

- Time based preview (time travel)

When a preview date is set in CoreMedia Studio, it sets the virtual render time to a time in the future. If the currently previewed page contains content from

*Hybris Commerce*, it is desirable that also these content reflects the given preview time. That could be a certain validity period of a product a another display rule that influences the displayed catalog items.

If such preview is requested from Hybris Commerce the preview date is also sent to Hybris Commerce as part of the `cmsTicket` parameter. The Hybris Commerce recognizes the transmitted preview date and renders the shop content accordingly.

- Customer segment based preview

The feature segment based preview supports the creation of personalized content. In this case, content is shown depending on the membership in specific customer segments. In addition to the existing rules, you can define rules that are based on the belonging to customer segments that are maintained by the commerce system.

These commerce segments will be automatically integrated and appear in the chooser if you create a new rule in a personalized content. For a preview, editors can use test personas which are associated with specific customer segments.

The following figure shows an example where the test persona is female and has already been registered.

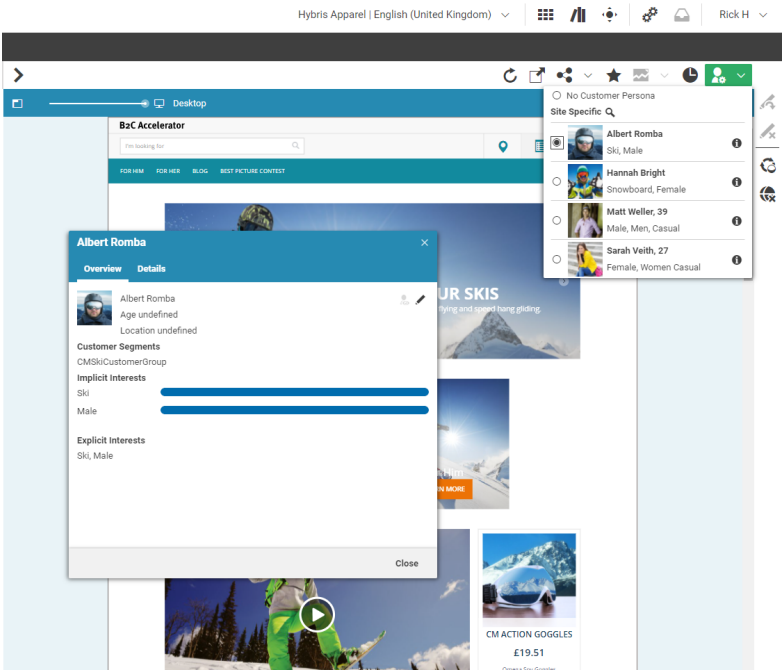


Figure 3.75. Test Customer Persona with Commerce Customer Segments

Such preview settings apply as long as they are not reset by the editor.

The test persona content item can be created and edited in *CoreMedia Studio*. The customer segments available for selection will be automatically read from the commerce system. By default all user segments available in the eCommerce system are displayed for selection. Under some circumstances it may be desirable to restrict the shown user segments, for instance for studio performance reasons or for better clarity for the editor. See ????

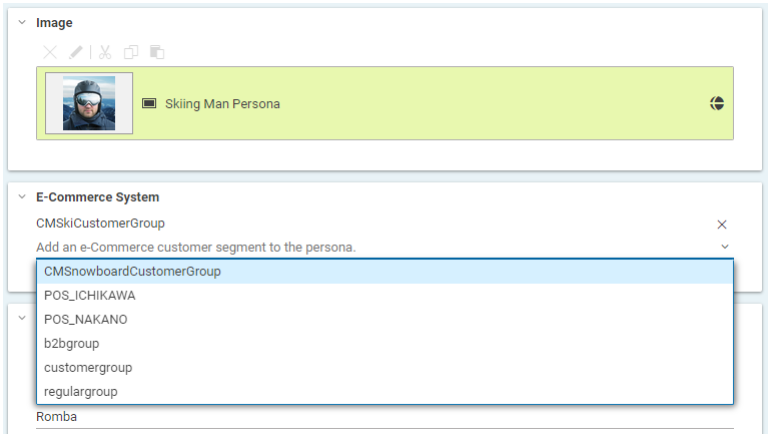


Figure 3.76. Edit Commerce Segments in Test Customer Persona

The commerce segments that the current user belongs to are available during the rendering process within a *CoreMedia CAE*. Thus, content from the CoreMedia system can also be filtered based on the current commerce segments.

In the other direction, if the personalized content is integrated within a content fragment on a shop page, the current commerce user is also transmitted as a parameter. Thus, the CoreMedia system can retrieve the connected customer segments from the commerce system in order to perform commerce segment personalization within the supplied content fragments.

## 3.3 CoreMedia Content Cloud for commercetools

CoreMedia Content Cloud integrates with *commercetools*. In the following it is simply called the "commerce system" or "the shop".

From classical shop pages, like a product catalog ordered by categories or product detail pages up to landing pages or homepages, all grades of mixing content with catalog items are conceivable. The approach followed in this chapter, assumes that items from the catalog will be linked or embedded without having stored these items in the CMS system. Catalog items will be linked typically and not imported.

### Catalog View in CoreMedia Studio Library

When the connection to a *commercetools* system and a concrete shop for a content site are configured, the *Studio* Library shows the commerce catalog to browse product categories and products in the commerce catalog and to search for products and product variants. After the editor has selected a preferred site with a valid store configuration the catalog view will be enabled and the catalog will be shown in the Library:

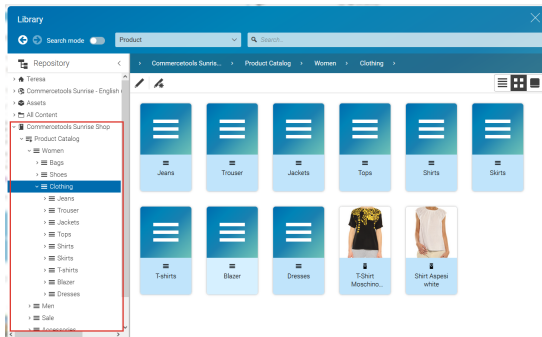


Figure 3.77. Library with catalog in the tree view

In some catalogs it is possible to put a category on multiple places within the catalog tree. But the Commerce Hub ensures that a category can only have one home (a unique parent category). All additional occurrences of a category are shown as a link in the tree. If you click on such a link node you will automatically end up at the place in the tree where the category is actually at home.

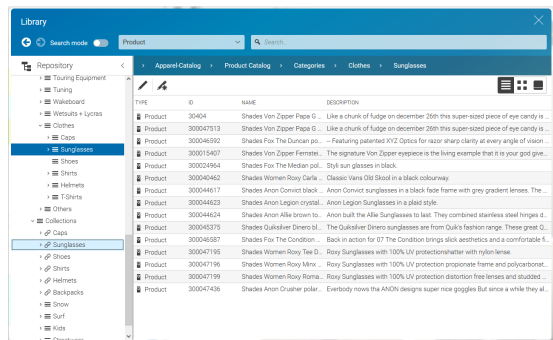


Figure 3.78. Library tree with multiple occurrences of the same category

These catalog items can be accessed and assigned to various places within your content. For example, an *eCommerce Product Teaser* content item can link to a product or product variant from the catalog. The product link field (in *eCommerce Product Teaser* content items) can be filled by drag and drop from the library in catalog mode.

Linking a content (like the *eCommerce Product Teaser*) to a catalog item leads to a link that is stored in the CMS content item and references the external element. Apart from the external reference (in the case of the commerce system it is typically a persistent identifier like the product code for products) no further data will be imported (importless integration).

While browsing through the catalog tree you can also open a preview of a category or a product from the library. Simply double-click on a product in the product list or use the context menu on a product or a category and choose the entry **Open in Tab** from the context menu as shown in the pictures below.

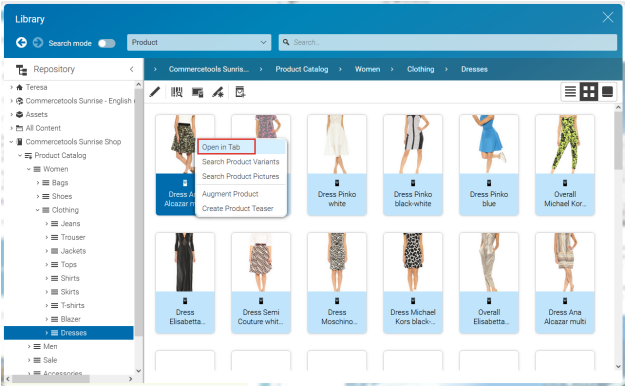


Figure 3.79. Open Product in tab

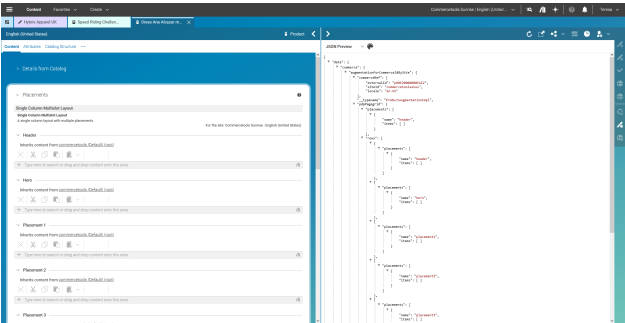


Figure 3.80. Product in tab with JSON preview

**NOTE**

For Information on how to enable the JSON preview have a look at [Section 9.34, “Multiple Previews Configuration”](#) in *Studio Developer Manual*.



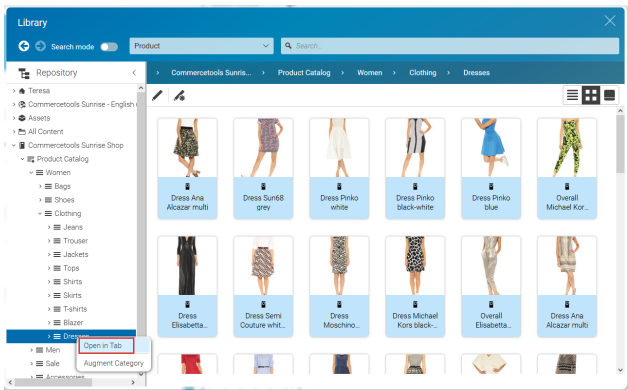


Figure 3.81. Open Category in tab

In addition to the ability to browse through the commerce catalog in an explorer-like view it is also possible to search for products and variants from catalog. As for the content search if you are in the catalog mode and you type a search keyword into the search field and press **Enter**, the search in the commerce system will be triggered and a search result displayed.



## 3.4 Custom Commerce Adapter

The *CoreMedia Commerce Hub* controls communication of CoreMedia apps with commerce systems by defining a vendor agnostic API covering the most common eCommerce features and providing a default client-server implementation of this API.

The client part of the *CoreMedia Commerce Hub* is named *generic client*. The server part is named *adapter service*. Adapter services are vendor specific extensions of the *base adapter* which itself defines the *Commerce Hub* API and serves as a runtime environment controlling the communication between generic client and commerce system.

- [Section 3.4.1, “Commerce Hub Architecture” \[59\]](#) describes the *Commerce Hub* architecture in more detail
- [Section 3.4.2, “Commerce Hub API” \[60\]](#) describes the APIs provided by the *Commerce Hub* and the request flow between *generic client*, *adapter service* and commerce system

### 3.4.1 Commerce Hub Architecture

Commerce Hub is the name for the CoreMedia concept which allows integrating different eCommerce systems against a stable API.

[Figure 3.82, “Architectural overview of the Commerce Hub” \[59\]](#) gives a rough overview of the architecture.

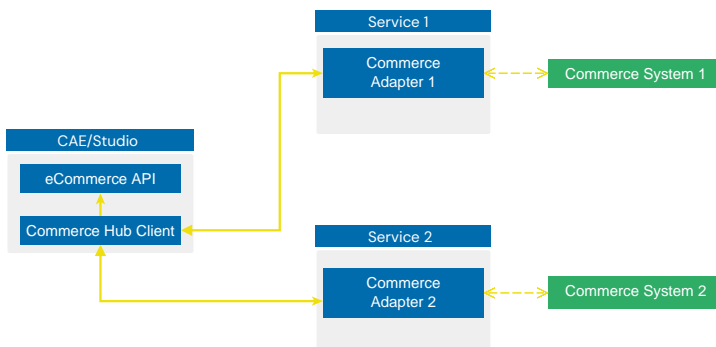


Figure 3.82. Architectural overview of the Commerce Hub

All CoreMedia components (CAE, Studio) that need access to the commerce system include a generic Commerce Hub Client. The client implements the CoreMedia eCommerce API. Therefore, you have a single, manufacturer independent API on CoreMedia side, for access to the commerce system.

The commerce system specific part exists in a service with the commerce system specific connector. The connector uses the API of the commerce system (often REST) to get the commerce data. In contrast, the generic Commerce Hub client and the Commerce Connector use gRPC for communication (see <https://grpc.io/>) for details.

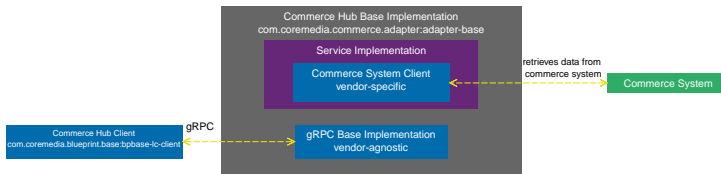


Figure 3.83. More detailed architecture view

Figure 3.83, “More detailed architecture view” [60] shows the architecture in more detail. At the Commerce Hub Client, you only have to configure the URL of the service and some other options, while at the Commerce System Client, you have to configure the commerce system endpoints, cache sizes and some more features.

## 3.4.2 Commerce Hub API

The *Commerce Hub* API consists of a gRPC API used by the *generic client*, and a Java API which consists of the Entities API as a wrapper around the gRPC messages, and a Java Feature API, used by the specific *adapter services*.

### The gRPC API

The gRPC API defines the messages and services used for the gRPC communication between *generic client* and *adapter service*. It is not necessary to access this API from any custom code. Access should be encapsulated, using the provided Java APIs, described below. In case the existing feature set does not fulfill all needs for a custom commerce integration, the gRPC API may be extended. CoreMedia provides two sample modules, showing a gRPC API extension in the *Commerce Adapter Mock*. Please have a look at the [Section 3.2, “CoreMedia Commerce Adapter Mock”](#) in *Custom Commerce Adapter Developer Manual*.

### NOTE

By Default the *base adapter* exposes the gRPC *ServerReflection* service. It is used by the *CoreMedia Commerce Hub Client* to obtain available features.



## The Java API

The Java API consists of two parts. The first part defines Java Entities as a wrapper around gRPC. It is used by the *generic client* and the server in the *base adapter*.

The second part is meant for server side only. It defines the Java Interfaces, called Repositories, the *adapter services* may implement for any needed feature. This API should be used as an entry point for commerce adapter development.

## Request flow

The request flow, using the above described APIs, starting from the generic client is as follows. Please have a look at [Figure 3.83, "More detailed architecture view" \[60\]](#) first.

1. The generic client sends a gRPC request to the vendor agnostic *base adapter*. The Entities API is used to convert the Java entity to the corresponding gRPC message.
2. The gRPC service implementation in the *base adapter* receives the gRPC request and invokes the corresponding repository methods.

While the API definition of the repositories is placed in the *base adapter*, the implementation which is called here is part of a specific commerce adapter.

The commerce adapter uses its vendor specific implementation to obtain the requested data from the commerce system. The data is then mapped to a CoreMedia commerce entity as defined by the base adapter.

Finally, the service implementation in the *base adapter* converts the given entity back to a gRPC response and sends it back to the *generic client*.

3. The *generic client* receives the gRPC response and uses the Entities API to obtain and process the requested entity.

## 3.4.3 Integrating a Custom Commerce System

As described in [Section 3.4.1, “Commerce Hub Architecture” \[59\]](#) the CoreMedia Commerce Hub consists of three main parts: a *base adapter* implementation defining the API and handling the request flow, a commerce system agnostic *generic client* implementation and a commerce system specific *adapter service*. In order to integrate a custom commerce system into the CoreMedia system, an *adapter service* for that system has to be implemented, using the *base adapter*.

### 3.4.3.1 Developing a Custom Commerce Adapter

An *adapter service* is the link between the generic CoreMedia eCommerce client and the specific commerce system. The following chapter shows how to get started, implementing a custom *adapter service*.

As described in [Section 3.4.2, “Commerce Hub API” \[60\]](#), the CoreMedia eCommerce API is defined in the *base adapter*. It offers a rich set of commerce features which can be used by implementing the corresponding repository interfaces. In order to implement the API, the `com.coremedia.commerce.adapter:adapter-base` and `com.coremedia.commerce.adapter:adapter-api` dependencies have to be added to your project.

The `adapter-base` dependency includes the repository interfaces for all available features. They can be found in the `com.coremedia.commerce.adapter.base.repositories` package.

The `adapter-api` dependency includes the most common eCommerce entities like catalogs, categories and products. They can be found in the `com.coremedia.commerce.adapter.api.entities` package.

#### The minimum feature set

As mentioned before, the CoreMedia eCommerce API offers a superset of commerce features which are all implemented on the client side. The *adapter service* (server side), should of course only implement the repositories for the needed features. The client requires access to catalogs and categories for building the commerce connection. Also, products are considered a mandatory feature.

The following features are required for establishing a commerce connection:

- **Catalogs:** Implement `CatalogRepository`
- **Categories:** Implement `CategoryRepository`
- **Products:** Implement `ProductRepository`

Custom searches for arbitrary commerce entities can be implemented via `com.coremedia.commerce.adapter.base.repositories.ProductSearchSupport#search` by implementing `ProductRepository` on the adapter side.

### CAUTION

*Base adapter* releases up to 1.5 also require a `PriceRepository` implementation



## 3.4.3.2 CoreMedia Commerce Adapter Mock

CoreMedia provides a dedicated mock *adapter service* implementation for customers and partners as a GitHub repository. It is meant as an example of how to implement a custom *adapter service* and provides a fully functioning Spring Boot service. The service can be build via Maven and runs either as a plain Spring Boot app or inside a Docker container.

In case the used technologies are applicable, CoreMedia recommends to use this project as a starter for building a custom *adapter service*.

### Structure

The workspace can be found at <https://github.com/coremedia-contributions/commerce-adapter-mock> and includes a set of modules.

To get started developing a custom *adapter service* the following modules are needed.

- `adapter-mock-lib` This module holds a sample implementation of a custom *adapter service* which is used by the Spring Boot app from the `adapter-mock-app` module.

The repository implementations in this module should be adapted and serves as starting point for developing a custom commerce adapter.

- `adapter-mock-app` This module holds the ready to run Spring Boot app for the Mock Commerce adapter. The implementation sources are separated in the `adapter-mock-lib` module.

The following modules contain convenience configuration, tooling and sample code for extending the commerce API by custom gRPC services.

- `adapter-mock` This module holds the Docker setup. Using the `dockerfile-maven-plugin` it can be used to build a Docker image for the mock *adapter service*.
- `adapter-mock-custom` This module includes service customization samples for the mock *adapter service*. It is referenced as dependency in the `pom.xml` file of the `adapter-mock-app`.
- `adapter-mock-custom-grpc` This module holds a custom gRPC API definition which is then used by the services in the `adapter-mock-custom` module
- `workspace-config` This directory holds additional workspace configuration like the *IntelliJ IDEA* run configuration for the Spring Boot app.

### Using the Commerce Adapter Mock

The CoreMedia Commerce Adapter Mock is not only a sample, showing how to implement a custom commerce adapter, but can also be used as a starter project.

If you decide to use the project as a starter, just checkout the latest revision from GitHub and rename and reorganize the modules and repositories as it suits your project.

The entry point for developing a custom commerce adapter is the `adapter-mock-lib` module. It contains the `repositories` package, holding repository implementations for a broad feature set, including the mandatory implementations for `CatalogRepository`, `CategoryRepository` and `ProductRepository`.

Beside the `repositories` package you will find some more packages, containing samples for retrieving data, configuration or dealing with preview tokens. These packages are not needed for setting up a custom *adapter service*.

### NOTE

To get a better idea of how to develop an *adapter service* you can also have a look at the CoreMedia *adapter services* for Salesforce, SAP Commerce or HCL Commerce.

Useful features like caching, using the CoreMedia `Cache` or Monitoring with services like `Micrometer` should be considered crucial for your custom commerce adapter as well.

The latest version of the sources can be found on <https://repository.core-media.com>. Usages of the CoreMedia Cache can be found in the `com.core-media.commerce.adapter.sfcc.cache` package.



### 3.4.3.3 Integrating a Custom Commerce Adapter

In order to use the custom *adapter service* with the CoreMedia system. A minimum set of configuration and setup is needed.

#### Configuring the *adapter service* Endpoint

To enable the *generic client* to connect to a custom *adapter service* an endpoint for that service has to be added on the client side. This is done by configuring a `gRPC Spring channel`. Use the default channel configuration for properties that apply to all services. The specific configurations are done per named channel, where the name is an ops-friendly string of your choice, such as `fooService`. This name is used to identify the service in the site's `commerce` settings struct's `endpointName` property.

Please also refer to the Javadoc of the method `com.coremedia.blueprint.base.livecontext.client.settings.CommerceSettings#getEndpointName()`

#### The Vendor Name

To integrate an *adapter service* with the CoreMedia system, a vendor name for the commerce system has to be configured via `metadata.vendor` in the *adapter service*. This name is used as a prefix for all commerce IDs by the coreMedia system and should therefore never be changed.

### The Commerce Settings

The *CoreMedia Commerce Hub generic client* expects a commerce system to have at least one catalog and a root category. If this is the case, no further configuration is needed to set up the commerce connection. If the commerce system provides multiple catalogs or stores, both may be configured via the site's commerce settings content item.

#### CAUTION

The commerce connection is an instance of the `GenericCommerceConnection` managed by the *generic client*. It is valid only if the *generic client* is able to create an instance of the `GenericStoreContext` while communicating with the custom *adapter service*.



After the commerce connection for the *adapter service* is set up correctly, the catalog along with its categories and products can be displayed in the *CoreMedia Studio* library.

#### NOTE

If the CAE is used for augmenting the commerce storefront the `LinkRepository` needs to be implemented.





## 4. Asset Management

For a website, images are required in different sizes and formats. For example, teaser need a small image with an aspect ratio of 1:1 in the sidebar and an aspect ratio of 4:3 in the main section. Images in articles and galleries are shown in 5:2 or 4:3 with a large size. And even these sizes are different on mobile devices and desktop displays.

### MIME Type Mapping

When building links to image variants in the CAE, the MIME type of the original image is used by default to determine the file extension of the links. To adjust these MIME types you can provide a mapping of original MIME types to desired MIME types in the setting `linkMimeTypeMapping`. The struct `linkMimeTypeMapping` contains String properties where the key is the MIME type of the original image and the value is the desired MIME type for the links to variants of this image.

You could for instance add this setting to the Responsive Image Settings content item next to the `responsiveImageSettings` struct like so:

```
<Struct xmlns="http://www.coremedia.com/2008/struct">
  <StructProperty Name="linkMimeTypeMapping">
    <Struct>
      <StringProperty Name="image/jpeg">image/png</StringProperty>
      <StringProperty Name="image/gif">image/png</StringProperty>
    </Struct>
  </StructProperty>
  <StructProperty Name="responsiveImageSettings">
    <Struct>
      <StructProperty Name="portrait_ratio1x1">
        ...
      </StructProperty>
      ...
    </Struct>
  </StructProperty>
</Struct>
```

With these settings all links to variants of images with MIME type `image/jpeg` or `image/gif` would be created with MIME type `image/png` and the file extension `.png` instead.

## 4.1 Multiple Images and Sizes

*CoreMedia Blueprint* supports different formats combined with different sizes. It comes with four predefined cropping definitions.

- portrait\_ratio3x4 (aspect ratio of 3:4)
- portrait\_ratio1x1 (aspect ratio of 1:1)
- landscape\_ratio4x3 (aspect ratio of 4:3)
- landscape\_ratio16x9 (aspect ratio of 16:9)

A list of sizes can be defined for each format in the *Responsive Image Settings*, located in the *Options/Settings/CMChannel* folder of the site. If no site specific setting is defined, the global setting *Responsive Image Settings* from */All Content/Settings/Options/Settings* will be taken. The website will automatically choose the best matching image depending of the viewport of the client's browser.

### High Resolution/Retina Images

*CoreMedia Blueprint* supports high resolution images. Set the BooleanProperty *enableRetinaImages* to true. If enabled, the JavaScript *jquery.coremedia.responsiveimages.js* is choosing a larger image according to the *devicePixelRatio* of the browser.

For Example the website wants to render an image with an aspect ratio of 4:3 and the best responsive image size is 400px : 300px. With a *devicePixelRatio* of 2, the JavaScript *jquery.coremedia.responsiveimages.js* is now choosing the size of 800px : 600px.

### Default JPEG Compression Quality

The default JPEG compression quality is 80% in *CoreMedia Blueprint*. This parameter is configured in *blueprint-handlers.xml* for the *transformedBlobHandler*. It can also be set per image variant in the *Responsive Image Settings* as described in [Section 9.5.3, "Image Cropping and Image Transformation"](#) in *Studio Developer Manual*. For further information consult the [Section 4.5, "Image Transformation API"](#) in *Content Application Developer Manual*.

## MIME Type Mapping

When building links to image variants in the CAE, the MIME type of the original image is used by default. The MIME type is used to determine the file extension of the links. To adjust these MIME types you can provide a mapping of original MIME types to desired MIME types in the setting `linkMimeTypeMapping`. The struct `linkMimeTypeMapping` contains String properties where the key is the MIME type of the original image and the value is the desired MIME type for the links to variants of this image.

You could for instance add this setting to the Responsive Image Settings content item next to the `responsiveImageSettings` struct like so:

```
<Struct xmlns="http://www.coremedia.com/2008/struct">
  <StructProperty Name="linkMimeTypeMapping">
    <Struct>
      <StringProperty Name="image/jpeg">image/png</StringProperty>
      <StringProperty Name="image/gif">image/png</StringProperty>
    </Struct>
  </StructProperty>
  <StructProperty Name="responsiveImageSettings">
    <Struct>
      <StructProperty Name="portrait_ratio1x1">
        ...
      </StructProperty>
      ...
    </Struct>
  </StructProperty>
</Struct>
```

With these settings all links to variants of images with MIME type `image/jpeg` or `image/gif` would be created with MIME type `image/png` and the file extension `.png` instead.

## 4.2 Working with Assets

*CoreMedia Advanced Asset Management* allows you to store and manage your digital assets (for example, high-resolution pictures of products) in the CoreMedia system. *CoreMedia Experience Platform* stores the asset in the original format (for example a PSD Photoshop file) together with several renditions for different use cases.

A rendition is a derivative of the raw asset, suitable for use in output channels, possibly with some further automated processing. A rendition might be, for example, a cropped and contrast-adjusted image in a standardized file format whereas the original file might be stored in the proprietary format of the image editing software in use. *Blueprint* contains a download portal, where you can comfortably present your assets for download.

*Definition of a rendition*

An `Asset` content item can not directly be used on your website, except for the download portal. Therefore, you can create a `Picture` or `Video` item from a rendition in an `Asset` item as described in [Section 4.2.3, “Creating Pictures from Assets” \[75\]](#) and [Section 4.2.4, “Creating Videos from Assets” \[77\]](#).

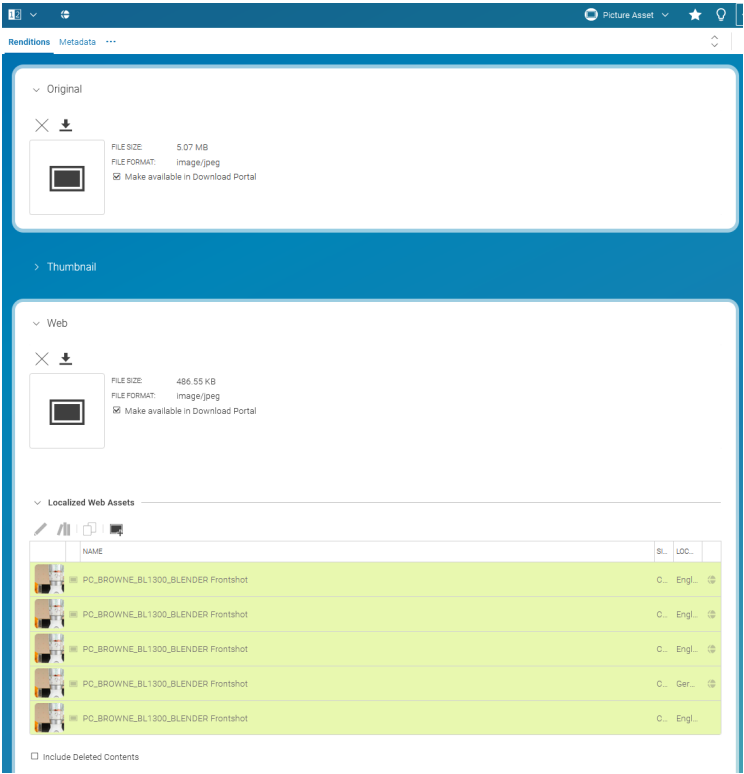


Figure 4.1. Picture Asset content item

## 4.2.1 Extracted Metadata

*CoreMedia Advanced Asset Management* extracts product codes from the IPTC metadata of JPEG pictures uploaded to **Picture Asset** items. You will find them on the **Metadata** tab in the **Product Code** field. If you create a **Picture** item from the asset, the **Picture** item will automatically be linked to the product with the corresponding product code.

### NOTE

By default, the CoreMedia system only reads in the *Source inventory number* field of the *Artwork or Object in the Image* section. Your implementation team can extend this by using the `XmpImageMetaDataAdapter` class, and the `ProductIdExtractor` class as a reference from the Blueprint.



A product code that can be read during the upload, has to be placed in the right field of the metadata. To do so, you can use any software that supports setting extended IPTC tags for the XMP standard, such as Photoshop.

*Adding metadata*

Add the product code to the *Source inventory number* field of the *Artwork or Object in the Image* section in the *IPTC Extension* tab (see [Figure 4.2, "IPTC metadata setting in Photoshop" \[73\]](#) for an example with Photoshop). You can add more than one product code to an image.

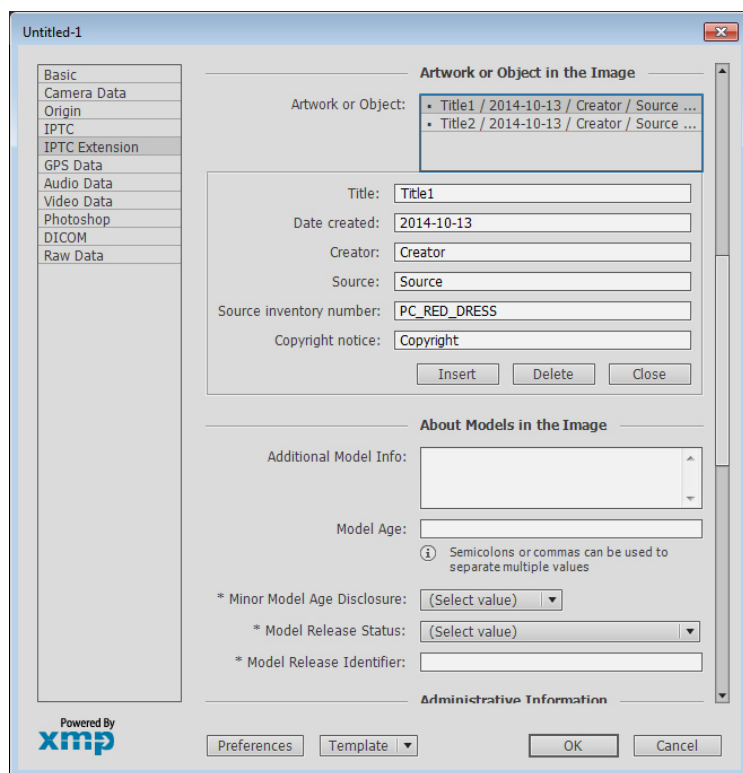


Figure 4.2. IPTC metadata setting in Photoshop

## 4.2.2 Creating Assets in Studio

You create assets in a freely configurable folder structure below the `Assets` folder.

In order to create an `Asset` content item, proceed as follows:

1. Select the folder where you want to create the asset, and create a content item of type `Picture Asset`, `Video Asset` or `Document Asset`.

The asset opens up and you can start editing.

2. Upload the original image or document and the renditions into the appropriate fields of the `Renditions` tab. The `Thumbnail` property is used for the preview, for instance in the library.

When your original picture contains a product ID in the XML IPTC metadata (see [Section 4.2.1, “Extracted Metadata” \[71\]](#) for details), then this ID is added to the metadata of the asset.

- 3. Enter your rights metadata into the *Rights* section of the *Metadata* tab. This information helps editorial staff to find assets that are licensed for their intended use.

In addition, the expiration date is used in the download portal to hide assets that are expired. The asset categories are used for the structure in the download portal and for searching. See [Section 4.2.5, “Categorizing Assets” \[78\]](#) for details.

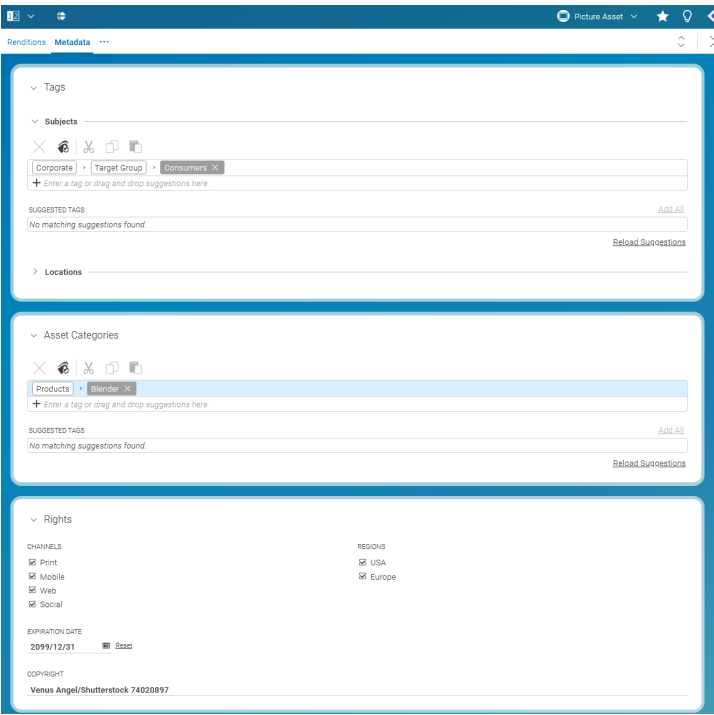


Figure 4.3. Metadata for Assets



## 4.2.3 Creating Pictures from Assets

In order to use a picture asset on your website, you first have to create a `Picture` content item, where you can add information required for presentation on a responsive, localized website.

1. Open the `Picture Asset` item and in the `Renditions` tab, open the collapsed field with the appropriate rendition. For example, `Web` for your website.

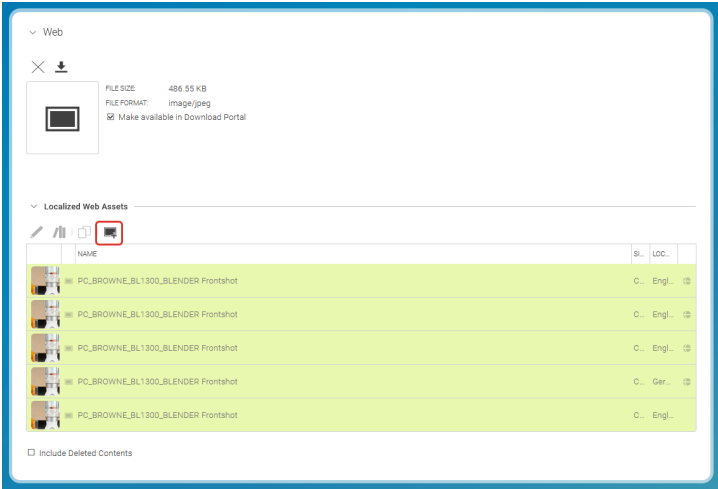



Figure 4.4. Create picture from asset button

2. In the `Localized Web Assets` field, click . A dialog opens up.

If you want, change the name or folder location, then click **[Create]**.

The newly created `Picture` item opens up and contains the rendition. You can edit it as usual. When the asset contains an expiration date, then this value is inserted into the `Valid to` property of the `Picture` content item. The copyright information from the asset is inserted into the corresponding property of the picture. The asset now shows the new picture in the `Localized Web Assets` field and the `Picture` content links to the asset.

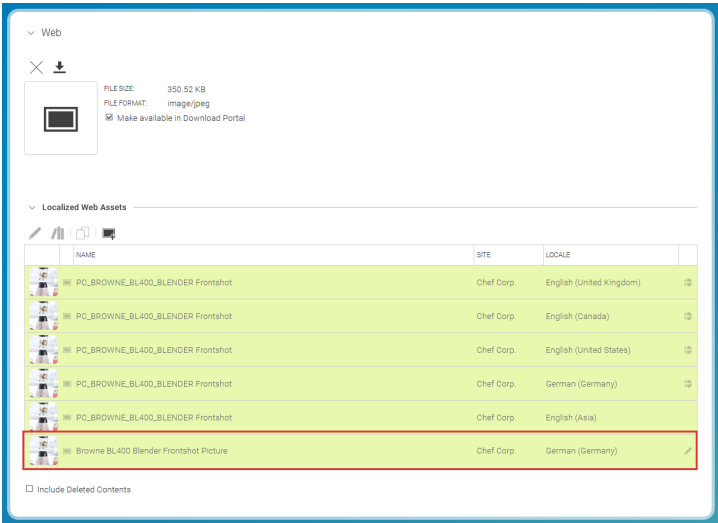


Figure 4.5. Newly created picture shown in Picture Asset

**Only for external catalogs:** If the Picture Asset item contains a Product Id in the *Metadata* field that corresponds to the ID of a product in the catalog, then the picture is automatically assigned to this product.

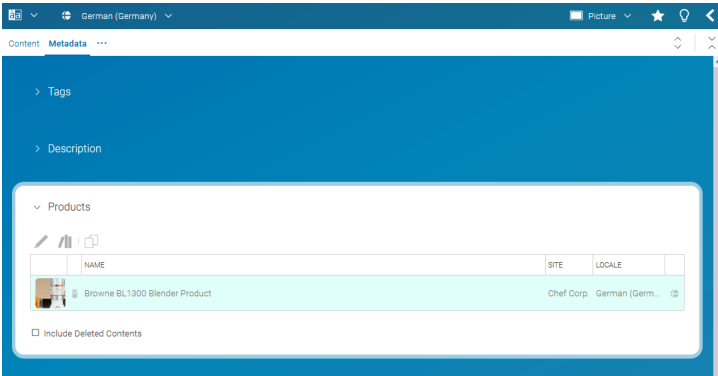


Figure 4.6. Product linked by picture

## 4.2.4 Creating Videos from Assets

In order to use a video asset on your website, you first have to create a `Video` content item, where you can add information required for presentation on a responsive, localized website.

1. Open the `Video Asset` item and in the *Renditions* tab, open the collapsed field with the appropriate rendition. For example, *Web* for your website.

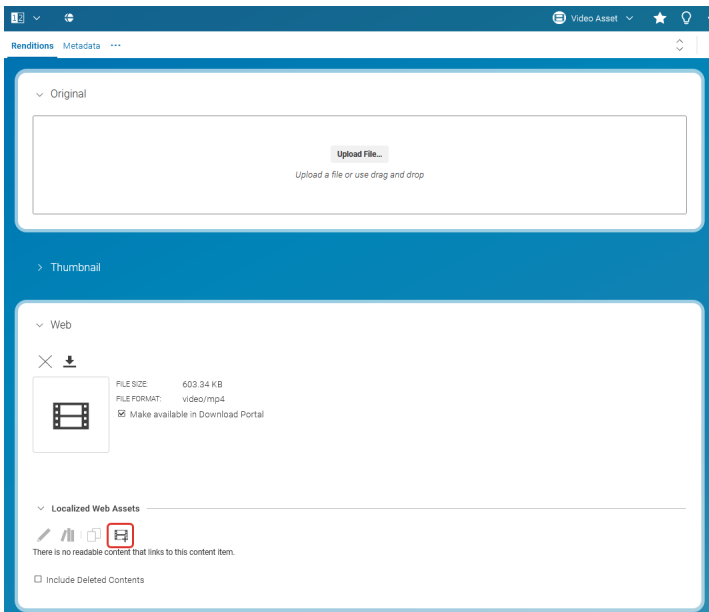



Figure 4.7. Create video from asset button

2. In the *Localized Web Assets* field, click . A dialog opens up.

If you want, change the name or folder location, then click **[Create]**.

For a video asset, a `Video` item and a `Picture` item are created. The picture contains the thumbnail for the video and is already linked from the *Pictures* field of the `Video` content item. The newly created items open up. You can edit it as usual.

When the asset contains an expiration date, then this value is inserted into the *Valid to* property of the `Video` content item. The copyright information

from the asset is inserted into the corresponding property of the Video. The asset now shows the new Video in the *Localized Web Assets* field and the Video content item links to the asset.

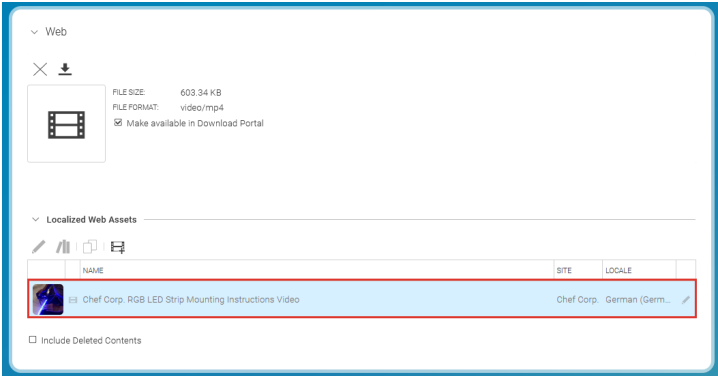


Figure 4.8. Newly created video shown in Video Asset

**Only for external catalogs:** If the Video Asset item contains a Product Id in the *Product Codes* field of the *Metadata* tab and the ID corresponds to the ID of a product in the catalog, then the picture is automatically assigned to this product.

## 4.2.5 Categorizing Assets

*CoreMedia Content Cloud* contains a specific taxonomy for assets based on Asset Category items. This taxonomy is used to create the download hierarchy in the download portal of *CoreMedia Experience Platform*.

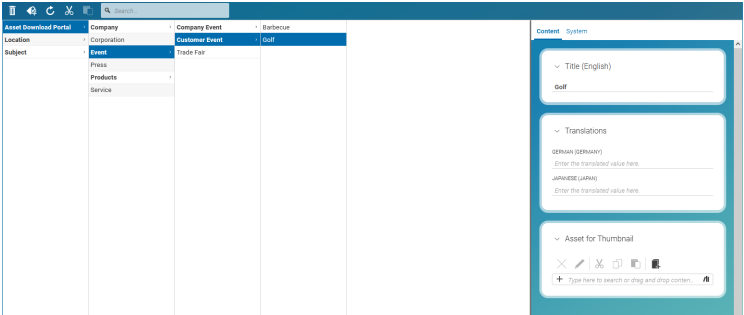


Figure 4.9. Taxonomy for assets

The taxonomy items are located in `All Content/Settings/Taxonomies/Asset Download Portal`. You can work with taxonomies as described in [Section 7.14.3, “Metadata Management” \[186\]](#).

### Adding a translation to a category

A category can have a translation for the other supported Studio UI languages. Simply add the translation to the corresponding field.

### Adding an image to a category

A category can have a thumbnail image that is used to represent the category in the download portal. In order to add an image, proceed as follows:

1. Select the category in the taxonomy editor.
2. Drag an `Picture Asset` content with a filled `Thumbnail` property onto the `Asset for Thumbnail` property of the category element.

### Categorizing an asset

You can simply select the category to which an asset belongs in the `Asset Categories` field.

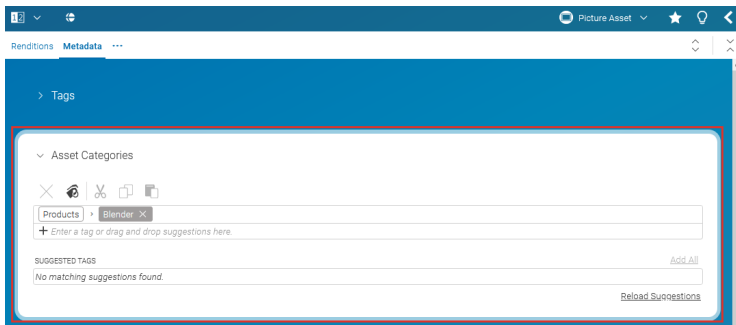


Figure 4.10. Add category to asset

You can use the category in searches and it is used to organize the download portal.

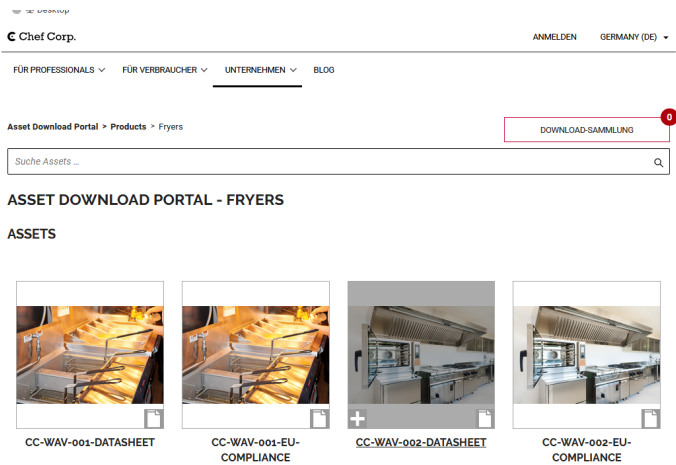


Figure 4.11. Download portal with Asset

## 4.2.6 Searching for Assets

You can search for assets below the `/Assets` folder, like you search for any other content. However, *Advanced Asset Management* enhances the *Filters* menu in the Library with asset specific filters and removes filters that are not applicable to assets.

<b>Asset Category</b>	Search for assets that belong to certain asset categories (see <a href="#">Section 4.2.5, “Categorizing Assets” [78]</a> ).
<b>Rights: Channels</b>	Search for assets for which you have rights for specific channels, such as web or print.
<b>Rights: Regions</b>	Search for assets for which you have rights for specific regions, such as USA or Europe.
<b>Expiration</b>	Search for assets for which your rights expire at a given point in time, for example in two weeks.



NOTE

Keep in mind that the filter options for one rights filter are jointly applied. That is, when you filter for channel rights with "web" and "print" selected, you will only get results which have at least rights for both channels.

## 4.2.7 Publishing Assets

You can publish an `Asset` content item like any other content. However, blobs in an asset might become very large, for example a PSD Photoshop file. Therefore, you can specify which properties of the asset should be published. The only exception is the *Thumbnail* property, which will always be published.

When you publish an asset, then all selected properties are shown in the download portal. To select a property for publication, simply click the *Make available in Download Portal* checkbox. If a property is empty, then no checkbox appears.

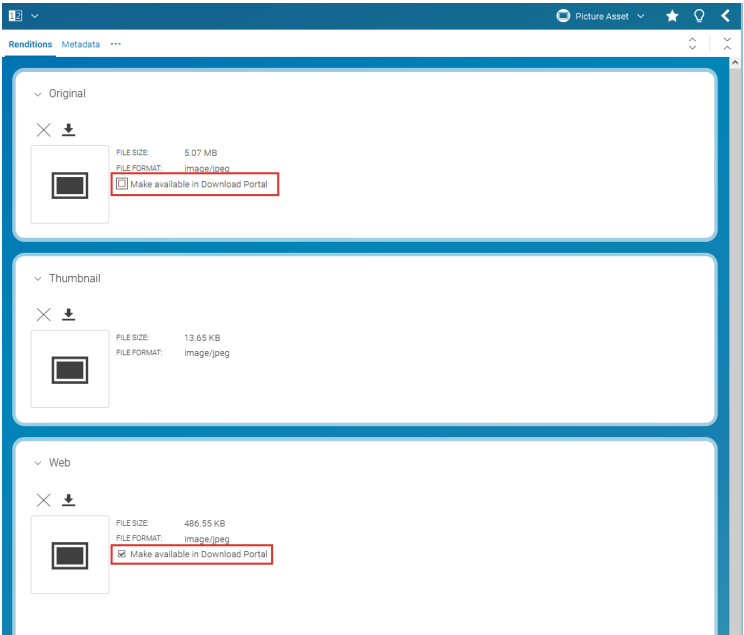


Figure 4.12. Select property for publication

## 4.2.8 Configuring the Asset Download Portal

*CoreMedia Advanced Asset Management* contains a download portal on the website.

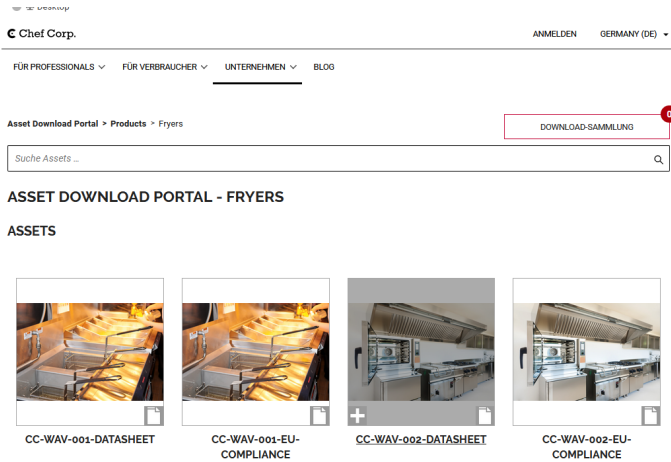


Figure 4.13. Asset download portal

You can configure the number of assets that are shown on one page of the portal. Proceed as follows:

1. Open the Asset Management Configuration item in *Studio*.
2. In the *Content* tab open the *Settings* field and adapt the *assets-per-page* property.



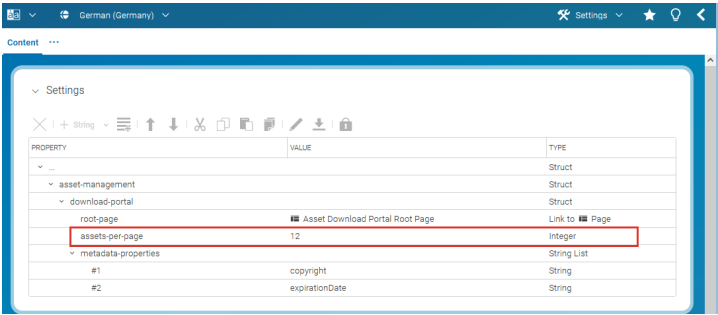


Figure 4.14. Configuration of the download portal

3. Save and publish the Asset Management Configuration content.

When you reload the preview of the download portal, you will see the new document limit at work.

# 4.3 Editing Images

*CoreMedia Studio* enables you to edit an uploaded image; even animated GIFs are supported.

You can, for example, flip, rotate, crop an image or change brightness and contrast. This functionality is non-destructive and can be reverted. You can use most of these functions like in your standard image processing software. Nevertheless, the cropping feature differs from the usual implementation that you know from other image processing tools.

The main idea in *CoreMedia Studio* is, that you need images for your website which have a fixed aspect ratio that fits perfectly into the page grid. Therefore, your *CoreMedia* system has been prepared so that you can choose different crops with a fixed aspect ratio and a minimal size for a given image. An example of possible crops are shown in the image . You can use the tabs in the form to select a specific crop. In the preview, you will see all crops in comparison.

So, for example, the 4:3 crop could be shown in an article on your website while the 1:1 crop is shown in a teaser. Where a crop is actually shown, depends on your specific *CoreMedia* system configuration.

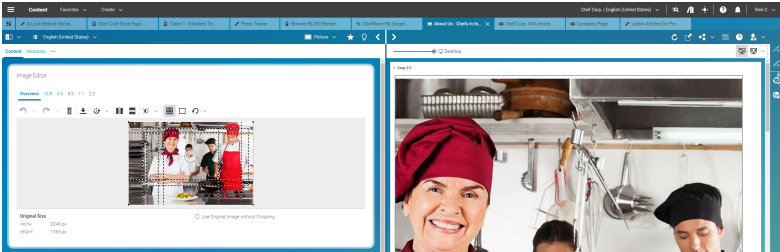


Figure 4.15. Cropping images

## The Selection Frame

*CoreMedia Studio* uses a specific concept to couple the selection frames of the crops in an image. When you have uploaded an image or opened an image content item, you will see all dashed selection frames in the overview.

By default, the complete image is selected and all frames have a common center in the middle of the image. The common selection area is highlighted by the solid white frame around the image. If you move or resize the common selection area, then all selection frames are moved and resized accordingly. If the common

*Common selection area*

area is smaller than the image, then the image outside the selection area appears darker.

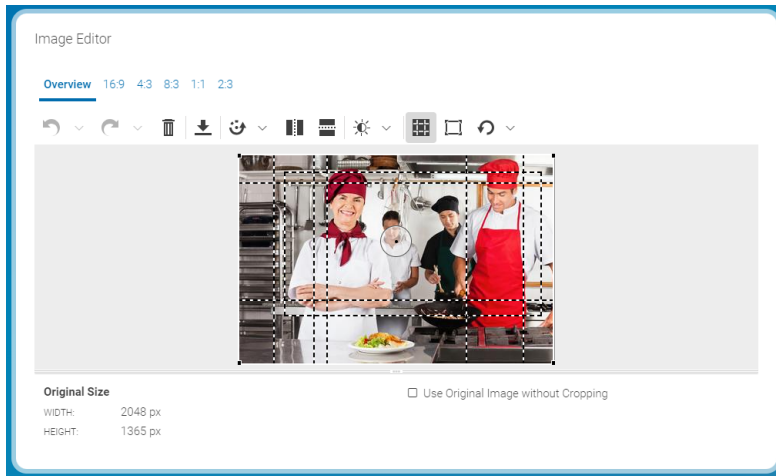


Figure 4.16. Overview with all selection frames

By default, all selection frames are centered around the center of the image. However, the center is not always the main point of interest in an image. So, for example, when you have a landscape image, with the main subject in the left side and you have a 1:1 crop, then the main subject might be missing in the crop. Therefore, you can move the focus point to another location. All selection frames in the common selection area of the image are rearranged in such a way, that the new focus point is as well as possible at the center of the selection frame.

*Arrange the focus point*

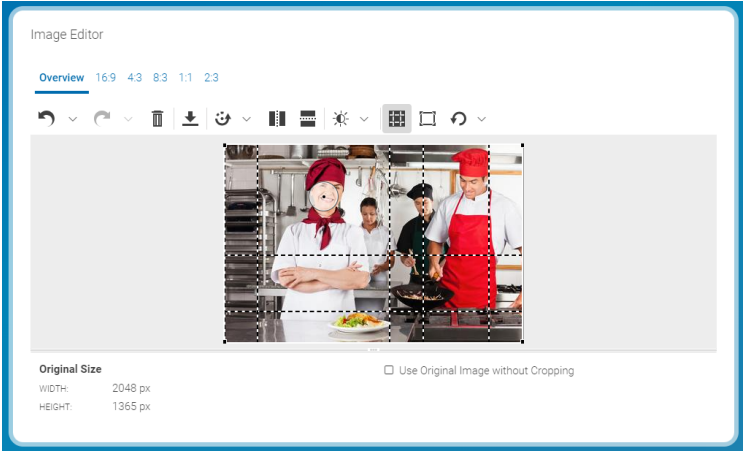


Figure 4.17. Moved focus point with automatically arranged 1:1 crop

If the common selection is not suited for a crop, you can decouple the crop from the common selection and edit it separately. You can check if a crop is decoupled by the position of the selection frame in the overview or in the separate view of a crop. A decoupled frame is not rearranged when you move the focus point

*Decoupling frames*

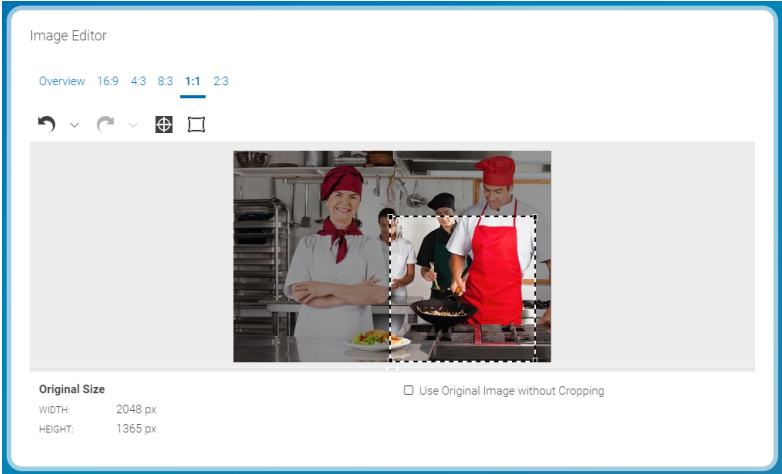


Figure 4.18. Decoupled selection frame

If a crop falls below the minimal width, its selection frame and label will be shown in orange and a warning message will be displayed in the list of content warnings

*Width warning*

and errors. You can still publish the content item, but you have to expect suboptimal image quality.

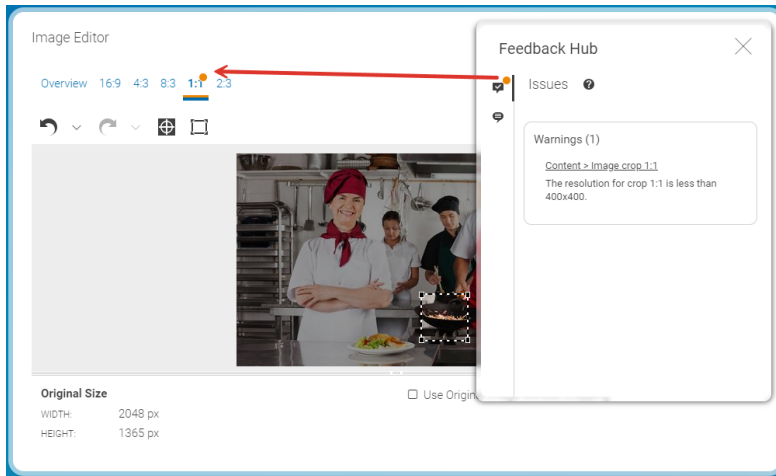


Figure 4.19. The crop is too small

## Cropping Images

CoreMedia Studio comes with different crops for an image. The size, aspect ratio and number of these crops has been defined by your project team while your CoreMedia system was implemented. Of course, these default setting can be changed, you should ask the person in charge for the CoreMedia system for support.

You can change the position and size – but not the ratio – of the crops.

When images are enlarged, the missing parts are filled with white color by default. This should be sufficient in most cases. However, you can define your own color.

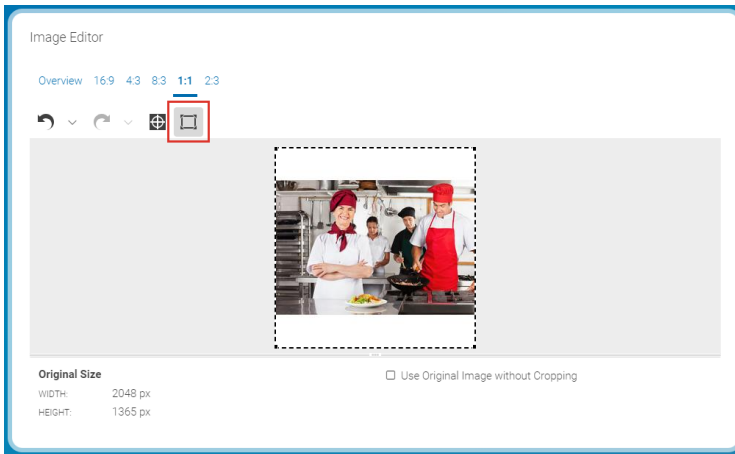


Figure 4.20. Enlarged image with white background

### Defining the background color

In order to change the background color used for the empty space of an image fitted to a crop, proceed as follows:

1. Open the *System* tab of the image and there the *Local Settings* field.
2. Add a String property named *background* to the Struct property *coloring*. If the property *coloring* does not exist, create it.
3. Add the background color as four hexadecimal byte values, for example FFO0FF00, to the *background* property. The first byte denotes the transparency (alpha channel) while the remaining three specify the red, green and blue color values, respectively

### Defining the common focus point

By default, all crops are centered around the center of the common selection area. However, you can move the focus point and Studio tries to center all selection frames around the focus point. The size of the selection frames does not change. Therefore, the selection frames will mostly not exactly center around the new focus point.

When you move or resize the common selection frame, then the focus point stays at its position until the common selection frame touches the focus point. When you proceed, the focus point follows the movement of the common selection frame.

In order to change the focus point simply move the focus icon to the new position.

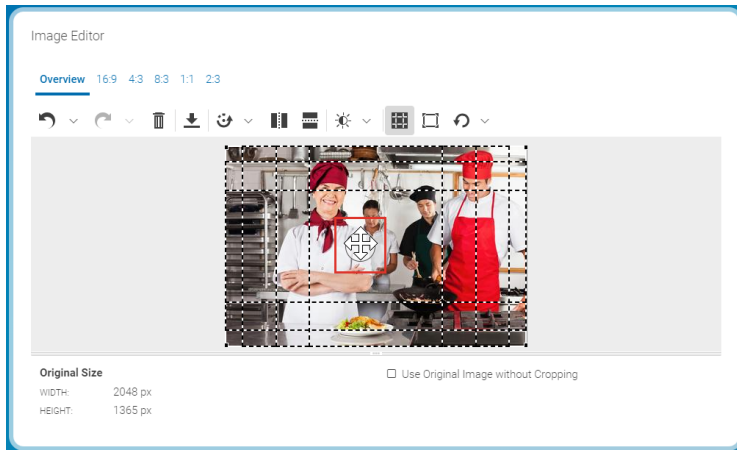


Figure 4.21. Move focus point

or

1. Choose the respective frame in the preview.
2. Right click on that image in the preview.

The selection frame is now shown without the other frames. Only the square icon indicates the center of the common cropping area.

## Flipping images

You can simply flip images at the horizontal or vertical axis with *CoreMedia Studio*. Keep in mind that always the whole picture is flipped. So, you cannot flip different crops of an image separately. The selection frames are also flipped, so that afterwards the same content is selected.

## Rotating images

You can either rotate images in *CoreMedia Studio* in 90° steps to the left and to the right or continuously align the position. Keep in mind that always the whole picture is rotated. So, you cannot rotate different crops of an image separately. However, *CoreMedia Studio* tries to keep the relative position of a selection frame with respect to the image. This can lead to an adapted size of the selection frames.

### Aligning images

In order to align a tilted horizon, for example, you can infinitely align an image in a given range. When you align the image, there would inevitably be corners with no content, therefore the image is zoomed in to avoid this. This adjustment keeps the aspect ratio of the selection frames, therefore the frames are not changed.

Select the *Rotate* menu and use the *Straighten* slider to align the image.

### Rotating images with 90° steps

Select the *Rotate* menu and click the *Rotate left* icon for a rotation to the left and the *Rotate right* icon for a rotation to the right. Click the *Reset* link to recreate the original position. Keep in mind, that the reset will not reset the original size and position of the selection frames.

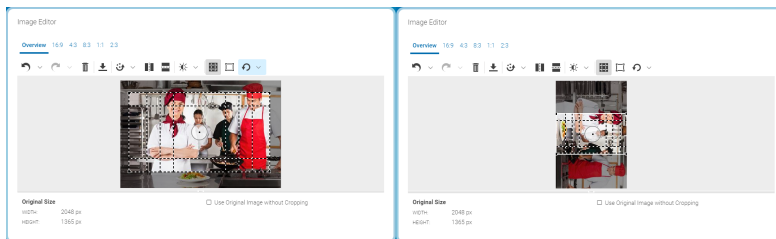


Figure 4.22. Effect of rotation on the selection frames

### Revert changes

If you want to recover the original orientation of the image, simply click the **[Reset]** button.

## Changing Exposure

*CoreMedia Studio* lets you adjust the exposure of an image to improve the quality.



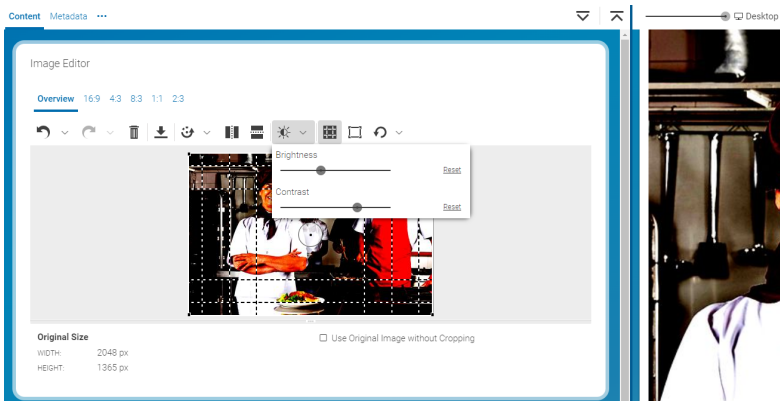


Figure 4.23. Exposure controls

### Adjusting Brightness

1. Click the *Change Exposure* icon.
2. Adjust the brightness with the *Brightness* slider to your needs.

Upon letting go of the slider, you will see the changed brightness immediately in the form and with a slight delay in the Preview. Using the **[Reset]** button, you can restore the initial brightness of your image.

### Adjusting Contrast

1. Click the *Change Exposure* icon.
2. Adjust the contrast with the *Contrast* slider to your needs.

Upon letting go of the slider, you will see the changed contrast immediately in the form and with a slight delay in the Preview. Using the **[Reset]** button, you can restore the initial contrast of your image.

## Reverting Changes

In *CoreMedia Studio*, you can undo a specific number of processing steps or redo steps you have previously undone. You can also undo all steps at once.

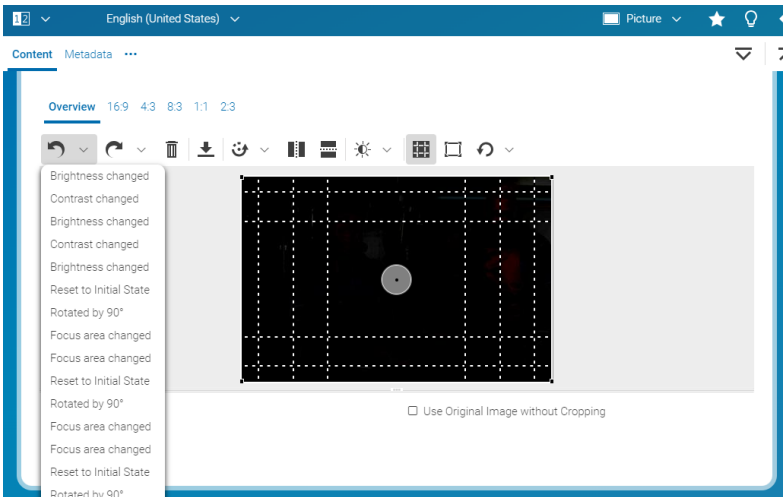


Figure 4.24. Undo steps

## 4.4 Editing Image Maps

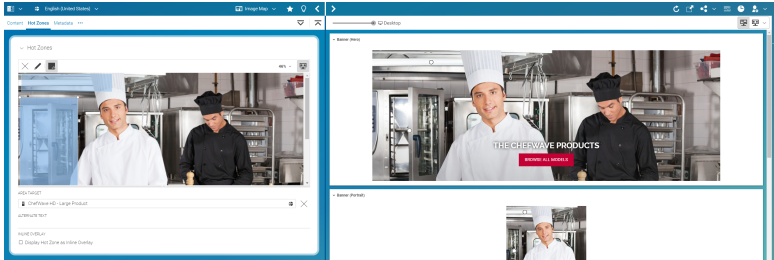


Figure 4.25. Hot Zones in an image map (pop-up)

CoreMedia Studio enables you to create and edit interactive image maps. Interactive image maps are made up of hot zones (image map areas) that are drawn on a picture and link to some target content. Depending on the hot zone configuration, the linked content is displayed in a pop-up close to the hot zone whenever a website visitor clicks on the hot zone or it is displayed as inline overlay which is always visible but only shows title and price information. You can configure several content properties, for example title or price that are shown in the overlay.

The following image shows the configuration options for image maps.

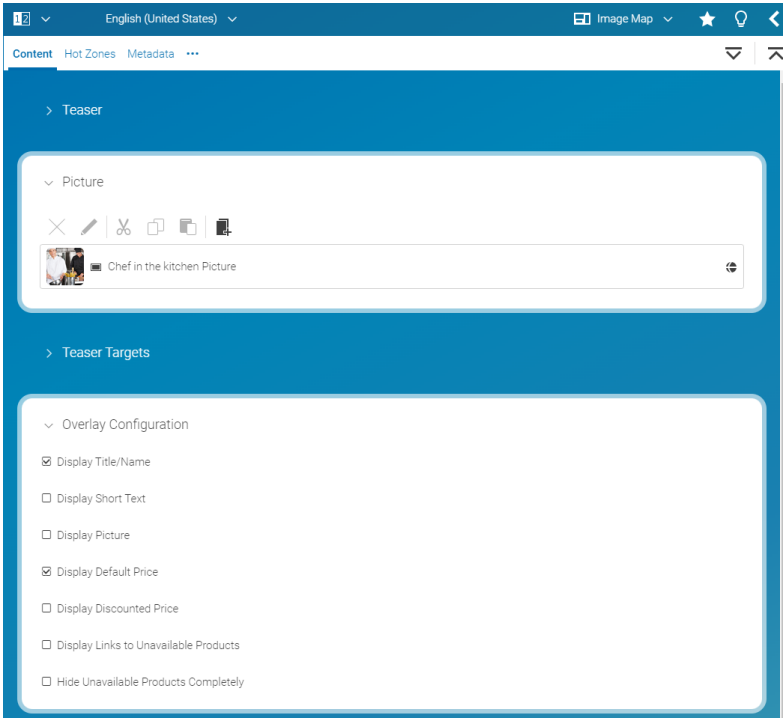


Figure 4.26. Configuring image maps

The hot zone overlay configuration is defined per image map and applies to all hot zones. For *Inline Overlays* the overlay configuration options *Display Short Text* and *Display Picture* are ignored.

## 4.5 Advanced Teaser Management

With Advanced Teaser Management *CoreMedia Experience Platform* allows you to manage text and call-to-action buttons on a teaser image.

### 4.5.1 Positioning Text on Teasers

For specific views of certain content types, such as the "Hero Teaser" view of an article or teaser, you can freely position and style an overlay containing the teaser text. If you have added a call-to-action button, this button will be positioned inside the overlay together with the text.

1. In the *Content* tab of the content item open the *Teaser* panel.

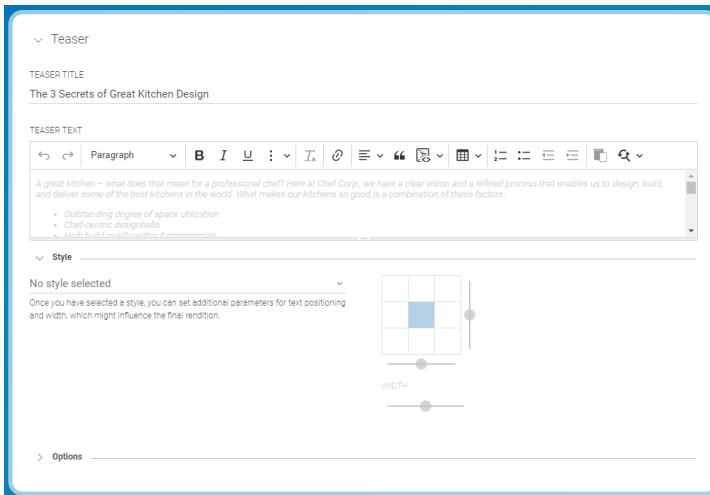


Figure 4.27. Open Teaser panel

Only the content of the *Teaser Text* field is used for the movable text. The content of the *Teaser Title* field is omitted.

2. Expand the *Style* section of the *Teaser* panel to select a global style for the text. The style defines the appearance of the overlay and its inner elements such as the font color and background.

Your available styles may differ from the styles in the image below.

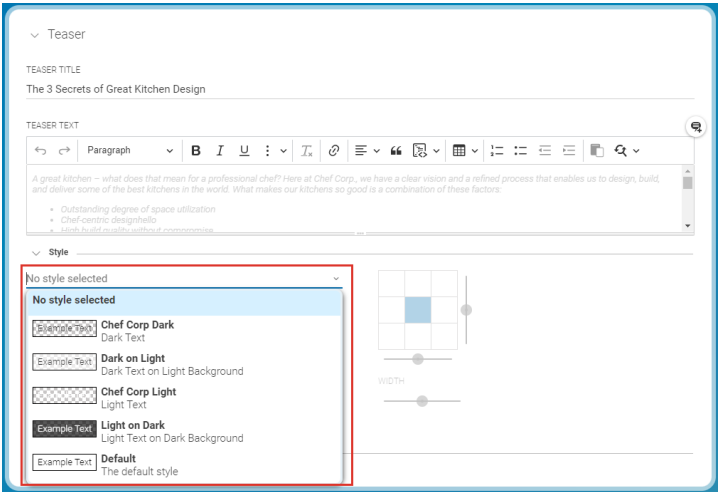


Figure 4.28. Select style for movable teaser text

3. Enter your text in the text field (1) and use the buttons in the toolbar (2) to format your text. You can choose a style, format the text as bold or italic and select the text alignment per paragraph.

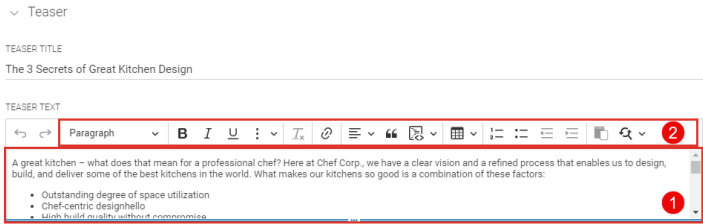


Figure 4.29. Formatting text

4. Expand the Style section of the Teaser panel to see the position menu. First, you have to select a style to activate the position menu. You can either click on one of the squares in field (1) to position the overlay at the corresponding location in the image or use the sliders (2+3) to position the text more precisely. With the Width slider (4), adapt the width of the overlay.

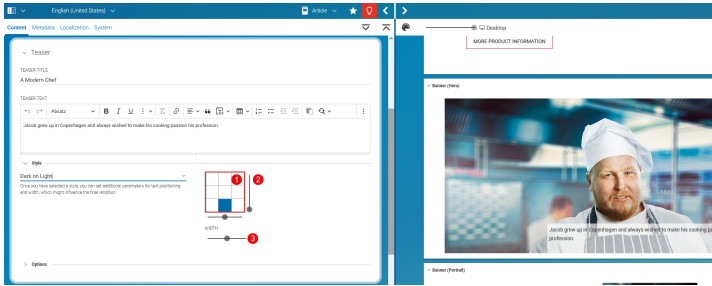


Figure 4.30. Position text

5. When you want to link from the teaser to the detail page of the content, select the *Render Link To Detail Page* checkbox (3).

## 4.5.2 Creating Styles for Teaser Overlays

The list of teaser overlay styles shown in the *Style* dropdown box is aggregated from styles found in `/Settings/Options/Settings/Teaser Styles` and `<SiteName>/Options/Settings/Teaser Styles`. A teaser overlay style is configured in a *CMSSetting* content.

## 4.5.3 Setting a Call-to-action Button

For a teasable content, you might define one or more (only for *Teaser* content) call-to-action buttons. That is a button shown in the teaser view of the content. By default, you will see a **[Learn More]** button. When a user clicks the button, then the whole content will be shown. The location of the call-to-action checkbox is different for teasable content (such as *Article*) and for *Teaser* items and inheriting items (such as *Image Map*).

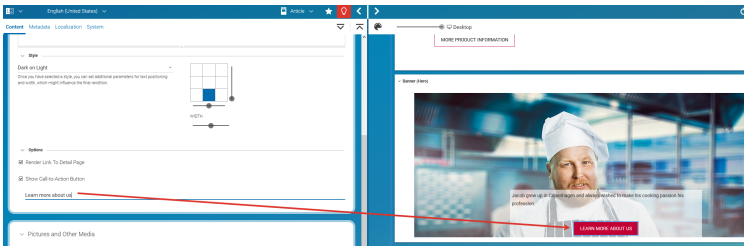


Figure 4.31. Call-to-action button in teaser view

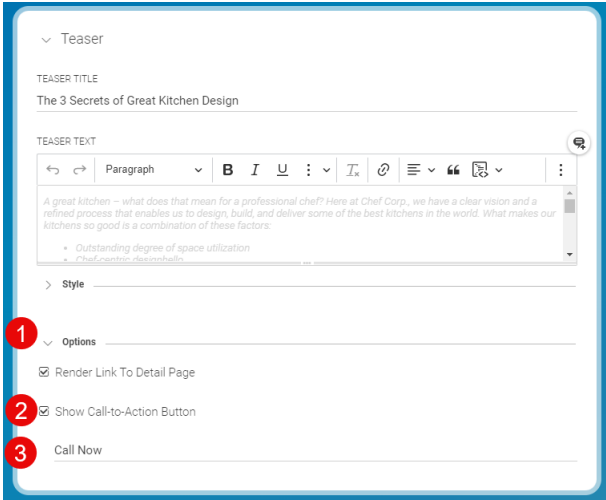


Figure 4.32. Call-to-action checkbox in teasable content item

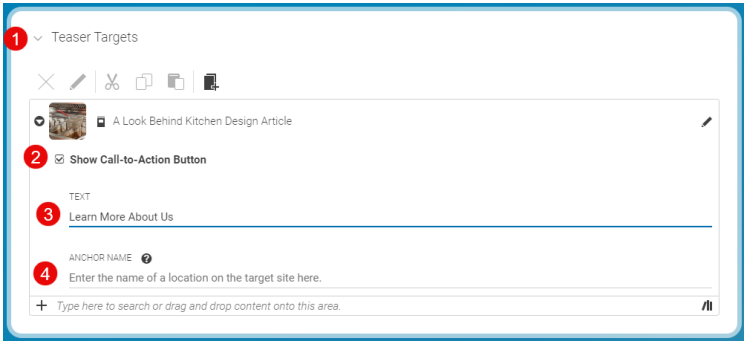


Figure 4.33. Call-to-action checkbox in Teaser content item




1. Open the *Content* tab of the content.
2. In the *Teaser* panel (for teasable content) open the *Options* field (1). For a Teaser content item open the *Teaser Targets* panel (1).
3. In the *Teaser Options* field (for teasable content) or *Teaser Targets* field, check the *Show Call-to-Action Button* checkbox (2).
4. If you want, enter a custom call-to-action text in the *Text* field (3). Otherwise, the default text will be shown.
5. For a Teaser content item, you can also add the name of an anchor of the target page, so that the call-to-action button leads to a specific section of the target content. Enter the plain anchor name (without the leading hash character), in the *Anchor Name* field (4).

You have to ask your theme developer for the available anchors of the target content items.

## 4.6 Editing 360°-Views

*CoreMedia Studio* enables you to create and edit 360°-Views. A 360°-View is a sequence of pictures which shows a product from different perspectives. On the website the 360°-View enables a user to rotate through the pictures by using touch controls.

Create a 360°-View content item by clicking the  button in the library. Edit the properties of the item in the form. Note, that a valid 360°-View must have more than two pictures. Link the 360°-View to a product, so that it will be shown on a product detail page.

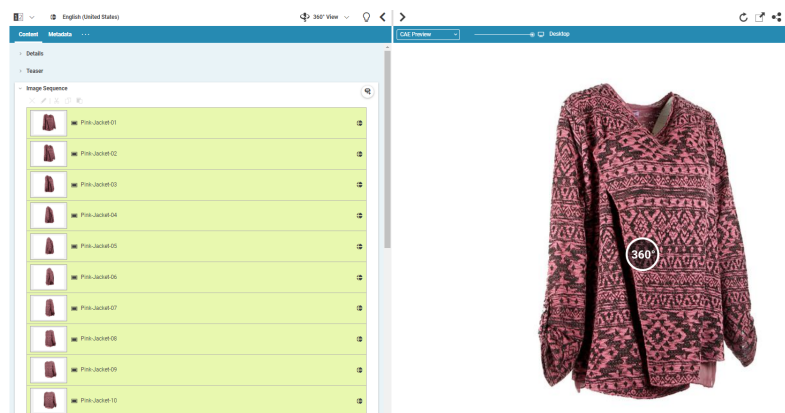


Figure 4.34. Images of a dress with a slight counterclockwise rotation

## 4.7 Editing Shoppable Videos

*CoreMedia Studio* enables you to manage videos in the Video content item. As a special feature, you can add content items that are shown synchronized with the video. For example, when the video shows a kitchen from second 40 to second 70, you can show cooking accessories beside the video in the same period while showing other products during the rest of the time.

This feature is intended to boost your sales by showing buyable products that match the content of the video. However, you can add all content items to the video that can be shown as a teaser, for example, Articles, Images, Collections and many more.

In order to add content items to a video, proceed as follows:

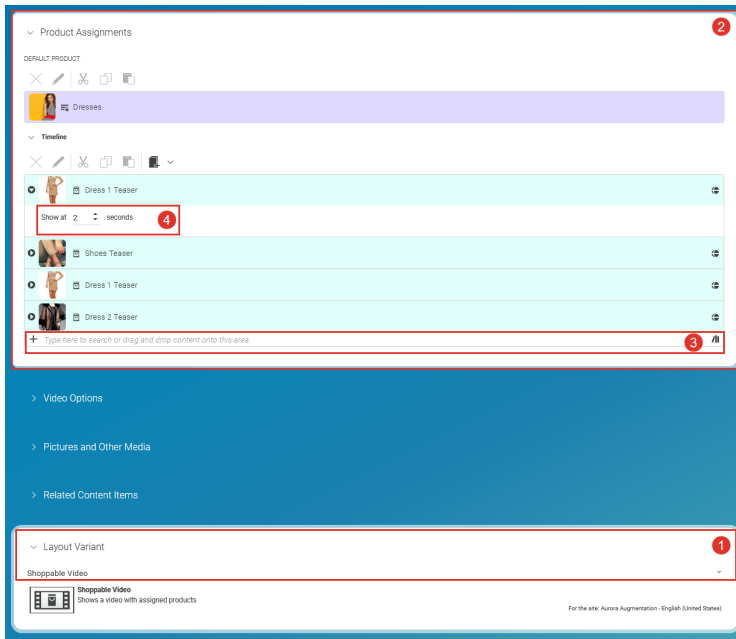


Figure 4.35. The shoppable video editor

1. In the *Content* tab of a Video open the *Layout Variant* panel and select *Shoppable Video* (1). The *Product Assignments* field (2) appears in the tab.

2. Drag a default product in the *Default Product* link list. The default product is shown until the first product appears on the timeline.
3. Drag other products in the *Timeline* link list (3).
4. Click the arrow to the left of the product in the timeline and enter the start time at which the product should be shown into the *Show at* box (4).

The product will be shown when the video reaches the specified point in time you have just specified. It will be hidden when the start time of another product in the timeline is reached. When no other product is in the timeline, then the product will be shown until the end of the video.

A validator warns when products in the timeline overlap.

Now you are done. The video is shown with the products beside.

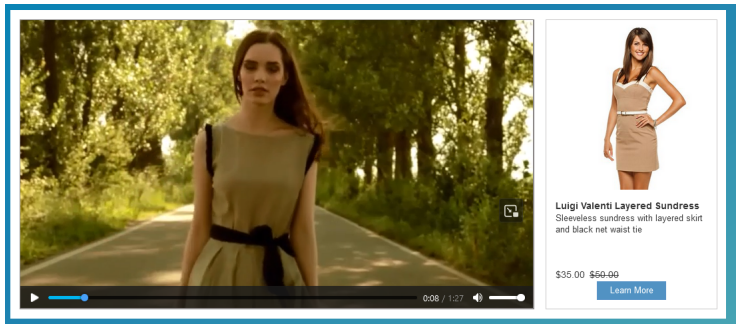


Figure 4.36. Site preview with shoppable video

# 5. Translation, Localization, Globalization

One of the primary challenges when engaging in a global market is to reach all customers in different countries.

The first most obvious task is to provide your website contents in different languages. But in addition, you may also want to customize your advertised products to local holidays or meet the different legal requirements in different countries.

CoreMedia's Multi-Site concept assists you in meeting these requirements.

There are many possible approaches to fulfill the requirements for providing multiple sites in different countries. CoreMedia Content Cloud offers a solution which you can customize to your needs and to the workflows you are used to.

The following chapter will present the basic ideas and concepts of CoreMedia's Multi-Site to you.

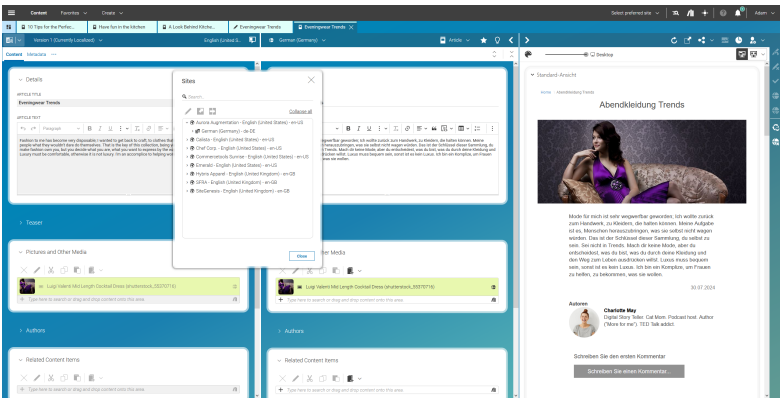


Figure 5.1. Side-by-side translation (review) within CoreMedia Studio

## 5.1 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.

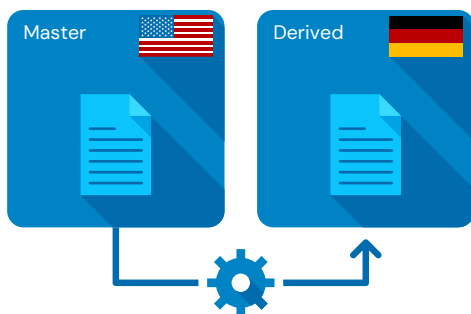
CoreMedia Multi-Site not only supports translation of content, but also localization, which is, that you may deliver sites having different languages and different structures. For example, you may want to add a banner for Thanksgiving holiday in your American site which is missing in all other sites.

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 5.2. 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.

## 5.1.1 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.

### 5.1.1.1 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.




## 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 “Variants of Site Hierarchies” \[106\]](#) 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 “Variants of Site Hierarchies” \[106\]](#) it denotes synchronized sites.

## Variants of Site Hierarchies

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



Language First (Recommended)

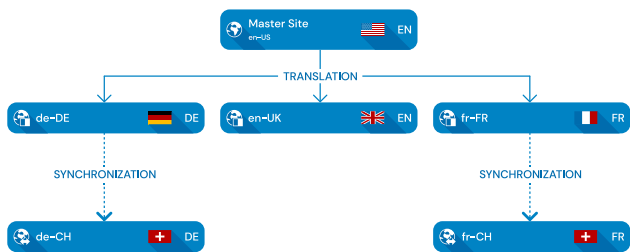


Figure 5.3. Sites: Language First

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” \[108\]](#) 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 5.3, “Sites: Language First” \[107\]](#) 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

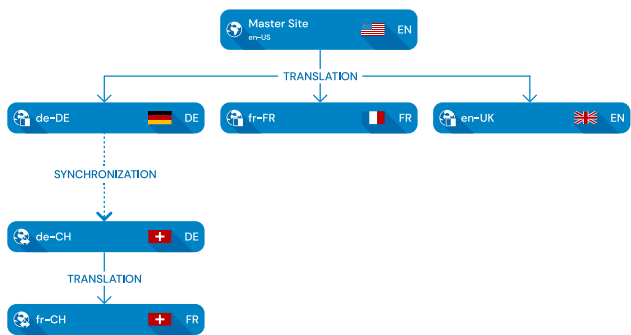


Figure 5.4. Sites: Country First

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 5.4, “Sites: Country First” [108] 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)*.

## 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.

## 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 5.3, “Sites: Language First” \[107\]](#) and [Figure 5.4, “Sites: Country First” \[108\]](#), 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.



# 5.2 Terms

The multi-site concept and documentation is based on the following terms. You may skip this section for now and return to it later when these terms are referenced.

Derived Site	A derived site is a <a href="#">site</a> , which receives updates from its <a href="#">master site</a> . A derived site might itself take the role of a master site for other derived sites.
Home Page	The site's home page is the main entry point for all visitors of a site. Technically it is also the main entry point to calculate the default layout and the contents of a site.
Localization	<p>In context of the multi-site concept, the main scope of sites within a <a href="#">web presence</a> is to provide localized versions of content items. This does not only include the translation of content items but also the adaptation of content items to the local culture, customs and traditions. The latter may result in, for example, other images to reference or just different content items to be referenced for a given <a href="#">locale</a>.</p> <p>In the context of <i>CoreMedia Blueprint</i>, the term <i>localization</i> often also serves as a generic term for the terms <a href="#">translation</a> and <a href="#">synchronization</a>. Thus, the term <i>localization workflow</i> refers to <i>translation workflows</i> as well as <i>synchronization workflows</i>.</p>
Locale	<p>A locale is briefly a combination of a language and a country or region. Within a <a href="#">web presence</a>, the site's locale must be unique.</p> <p><i>locale = language + country</i></p>
Master Site	<p>The locale is represented as IETF BCP 47 language tag (<i>Tags for Identifying Languages</i>).</p> <p><i>IETF BCP 47</i></p> <p>A master site is a <a href="#">site</a> other localized sites are derived from. A localized site might itself take the role of a master site for other <a href="#">derived sites</a>. This reflects the need that, for example, your</p>

	localized Canadian site (which is in English) needs another localized variant in French.
<b>Root–Master Site</b>	A root-master site is the top-level site in a multi-site hierarchy. The root-master site is the only site for a given <b>web presence</b> that is not derived from another site.
<b>Site</b>	<p>A site is a cohesive collection of web pages in a single locale, sometimes called localized site. Technically, a site consists of:</p> <ul style="list-style-type: none"><li>• The <b>site folder</b></li><li>• The <b>site indicator</b>,</li><li>• The <b>site's home page</b> and</li><li>• Other contents of the site.</li></ul>
<b>Site Folder</b>	All contents of a <b>site</b> are bundled in one dedicated folder. A typical example of a site folder is:
	<code>/Sites/MySite/Canada/French</code>
<b>Site Indicator</b>	<p>A site indicator is the central configuration object for a <b>site</b>. It is an instance of the content type <code>CMSite</code>. It explicitly configures:</p> <ul style="list-style-type: none"><li>• The <b>site's home page</b></li><li>• The <b>site ID</b></li><li>• The <b>site name</b></li><li>• The <b>site's locale</b></li><li>• The <b>master site</b></li><li>• The <b>site manager groups</b>.</li></ul> <p>It also implicitly defines the root of the <b>site folder</b>.</p>
<b>Site ID</b>	The site ID needs to be unique among all sites. It can be used to reference a site reliably also outside the CMS, for example, in configuration files.
<b>Site Manager Group</b>	<p>Members of a site manager group are typically responsible for one localized site. The recommendation is to have one dedicated group for each site with appropriate permissions applied for the <b>site folder</b>.</p> <p>Responsible means that they take care of the contents of that site. This includes but is not limited to:</p>

- adapting contents to local needs,
- accepting translation workflows, or
- triggering localization workflows, thus, synchronization or translation workflows to their site.

For the latter, the corresponding users need to be added to the translation manager role.

While the Site Manager Groups are typically local to their site, there is another role that is eligible to manage all sites within a [web presence](#). This role is called *global site manager*, while the other is referred to as *local site managers*.

*Global and Local Site Manager*

**Remark on permissions:** While local site managers are typically managing only the contents in their site, they still need read access to the master site to be able to manage translation processes or to compare their local contents with the master site contents.

**Site Name**

The site name is the name of a [master site](#) and all [derived sites](#). A derived site inherits the site name from its master site and must not change it.

**Synchronization**

Synchronization is a special type of [localization](#) process. It mostly means to provide one-to-one copies of content items for one site to a derived site with only minor adaptations to the derived content items.

A typical example of two sites that are synchronized is that both share the same language but are in different countries.

To synchronize content items, a built-in synchronization workflow is used, and the target derived site, that receives the synchronized content items, is referred to as *synchronization site*.

**Translation**

One element of the [localization](#) process is the translation of a content item from one [locale](#) to another.



The term refers to the actual translation done by editors or translation agencies as well as, in the context of *CoreMedia Blueprint*, to the translation workflow that guides editors through the translation process. The target derived site, which receives the translation results, is referred to as *translation site*.

### Translation Manager Role

Editors in the translation manager role are in charge of triggering localization workflows either from or to a site.

Different to what the name suggests, the translation manager role also applies to the ability to start synchronization workflows.

### Variants

The set of all content items related to each other via master references. This includes the top-level master content items themselves.

In typical multi-site setups this definition can be simplified to all content items that share the same root-master content item directly or transitively via master references.

### Web Presence

A web presence is a collection of all sites below the same [root-master site](#) — including the root-master site itself.

# 5.3 Sites Structure

CoreMedia Experience Platform organizes your sites for a given web presence in a strictly hierarchical, top-to-bottom structure starting with the root-master site. Connections between sibling nodes are unavailable and can only be expressed implicitly by a common ancestor.

The site hierarchy might be nested, thus a site derived from the master site again might have derivatives. You can trigger the localization process from your master site, directly derived sites will adapt and forward changes to their derived sites.

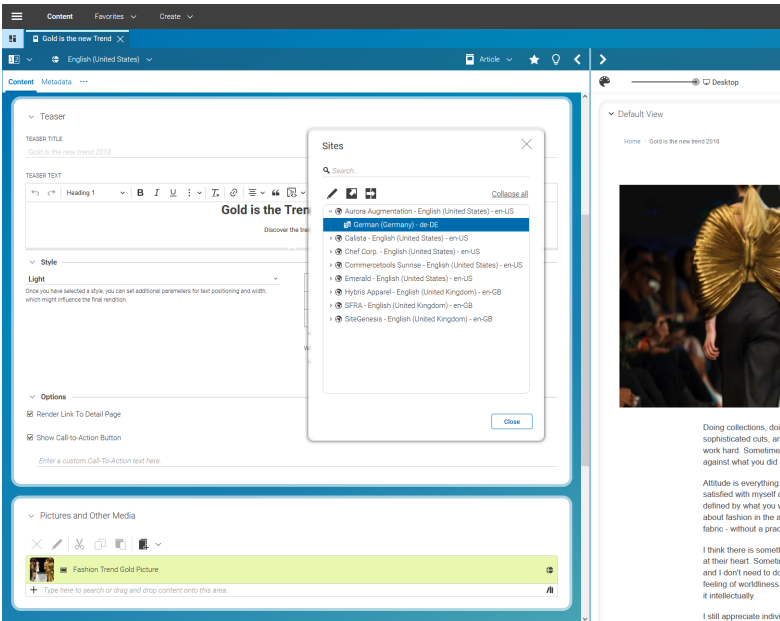


Figure 5.5. Derived Sites within CoreMedia

The examples below refer to the default configuration which comes with CoreMedia Blueprint.

## Multi-Site Folder Structure

All sites within a web presence share a dedicated root folder, that contains the root-master site as well as all its (nested) derived sites. Other web presences might be placed in parallel to this folder structure below the /Sites/ folder.

```
/Sites/  
  MySite/  
    United States/  
      English/ master  
      Spanish/ derived from U.S. English  
    Canada/  
      English/ derived from U.S. English  
      French/ derived from Canadian English  
  MyOtherSite/ another master site structure
```

### Example 5.1. Multi-Site Folder Structure Example

The folder structure of the root-master site and its derived sites is best kept equal to avoid the automatic recreation of removed or renamed folders during the localization workflows.

In addition to this, common aspects for all sites might be placed outside this folder structure.

## Site Folder Structure

It is the location of the site indicator that defines the site folder. Thus, a folder containing a valid site indicator with all required properties set (like the site ID) is regarded as a site folder. The site indicator also references (as root document) the site's home page.

The site's home page is placed in `Navigation`, so that the key content item structure is like this:

```
MySite/  
  United States/  
    English/  
      MySite [Site] site indicator  
      Navigation/  
        MySite site's home page  
      ...
```

### Example 5.2. Site Folder Structure Example

While the above describes the mandatory folder structure for a site, there are additional structures which adhere to the proposed separation of concerns, thus within a site you can have several user roles taking care of different aspects of the site as there are:

- **Editorial content:** For example, articles, images, collections etc. This is the real content of a site that is rendered to the web page. They are located in folders `Editorial`, `Pictures` and `Videos`.

- **Navigation content:** Channels that span the navigation tree and provide context information, as well as their page grids (see also [Section 7.3, “Navigation and Contexts” \[157\]](#)). These contents are located in a folder named `Navigation`.
- **Technical content:** Site specific, technical content items, like actions, settings, view types, etc. They can be found in folder named `Options`.

### Site Interdependence

Having a site derived from its master you will have two layers of interdependence:

1. The site indicator points to its master site indicator.
2. Each derived content item points to its master annotated by the version of the master when the derived content item retrieved its last update from the master. This information is used in the update process when a new master version requires its derived contents to be localized again.
3. A site indicator inherits the site name from its master. If a site indicator has no master it has to define the site name, which will be used for all derived sites.

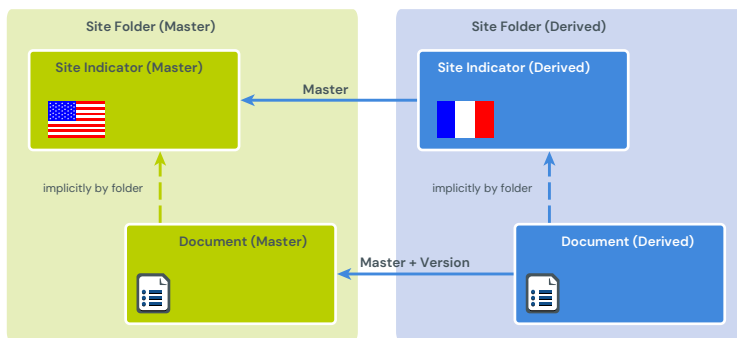


Figure 5.6. Multi-Site Interdependence

The `master` property is configured as weak link by default. Thus, you might publish derived sites before (or without) publishing the master site.

## Locales Administration

Each site is bound to a specific locale. In order to ensure a consistent usage of locale strings across multiple sites that might be managed in a single content

repository, the entire list of available locales is maintained in a central content item of type `CMSettings`.

Sometimes you might want to define locales for a supranational region such as Africa or Latin America. In this case you can add the language code followed by the UN M.49 area code as described in [https://en.wikipedia.org/wiki/UN\\_M.49](https://en.wikipedia.org/wiki/UN_M.49). For Spanish in Latin America and the Caribbean add, for example, "es-419".

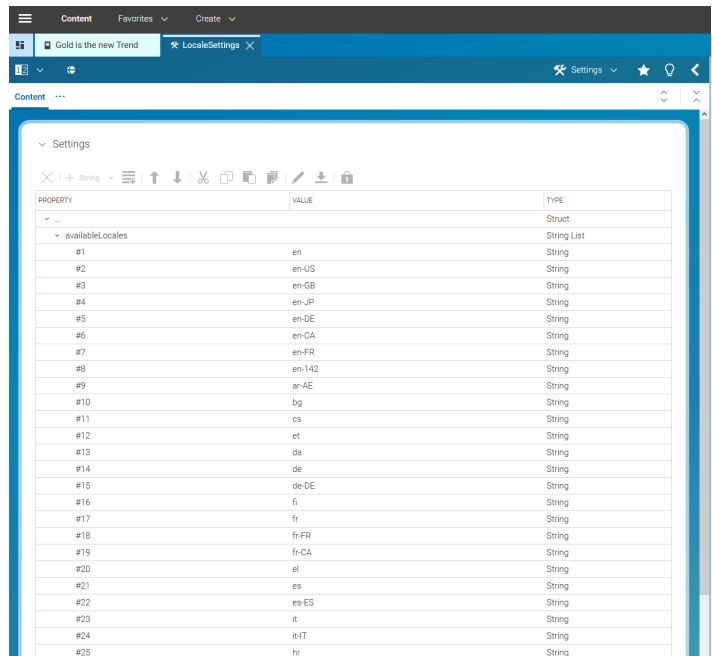


Figure 5.7. Locales Administration in CoreMedia Studio

# Groups and Rights Administration

This chapter describes all groups and users, that have to be defined for localization. There are several explicit groups and one user, that can be configured in the `SiteModel`.

The translation manager role is defined once in the property `translationManagerRole` of the `SiteModel`. It is a required group for every user that needs to start a localization workflow and to derive a site.

In case, you do not want to allow every translation manager to also derive sites, it is advisable to create an additional global site manager group, that has the right, to make modifications in the global sites folder.

Members of a site manager group take care of the contents of one or more sites. They may for example accept translation workflows if they manage the corresponding target site of a workflow. Or they may start a localization workflow (be it synchronization or translation) from the master site. For the latter, they must also be member of the translation manager role group, which is described above.

Derive a New Localized Site

Locale

English (Japan)

URL Segment

aurora-augmentation-en-jp

Site Manager Groups

manager-en-JP

Synchronization

☐ Synchronize with Master

OK

Cancel

Figure 5.8. Derive Site: Setting site manager group

If the site already exists, the names of site manager groups can be set or modified directly in the site indicator.

# 5.4 Translating Content

NOTE

CoreMedia offers an integration with GlobalLink Connect Cloud in CoreMedia Labs. If your company uses this integration, you can find the documentation on GitHub Pages at <https://coremedia.github.io/coremedia-globallink-connect-integration/>.



CoreMedia Content Cloud lets you manage sites in different languages. It supports the translation process with the side-by-side view, a translation workflow, export and import of content in the standard XML Localization Interchange File Format (XLIFF) and lets you derive an existing site for translation.

NOTE

In addition to these sections, also have a look into the [Multi-Site Manual](#). The manual describes different options to design your site hierarchy and gives some guidance to avoid common pitfalls when working with multi-site content.



When you translate a content, most properties that do not need translation will be updated to the content of the master property. For example, you have an integer property which contains the weight of a product and you change it in the master, then the weight will be updated in the derived site, when you start a translation workflow. However, this behavior can be changed in the content type definition. So, there might be properties which will not be updated.

*Properties that do not need translation*

Icon	Name
	Open sites window
	Derive site
	Translation to derived sites
	Translation to preferred site
	Download XLIFF


Icon	Name
	Compare content items

Table 5.1. Icons for Translation

## 5.4.1 Preparing Translation: Deriving a Translated Site

Deriving a new localized site means that you create a copy of an existing site. The content, structure and appearance of the new site is exactly like that of its master. Afterwards, you can start the translation of the derived content.

In order to derive a site, do the following:

1. In the Header Bar open the **Main** menu and select **Sites**. The sites window opens.

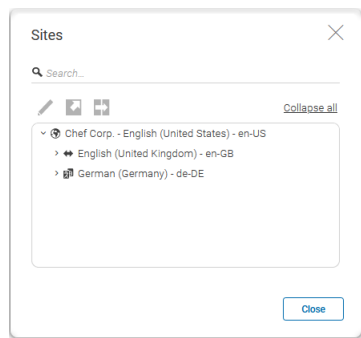


Figure 5.9. Sites Window

2. Select the site you want to derive from and click the *Derive a new Localized Site* icon. Alternatively you may use the context menu. A modal window opens up.



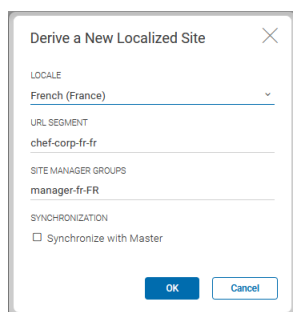


Figure 5.10. Derive a site dialog

- By default, the *Locale* field will be filled with the locale of the master site and will be marked red, because the locale has to be unique. Therefore, you have to choose a new locale first. Subsequently, new values for the URL segment and the site manager group will be calculated automatically using the selected locale. (You can add new locales to the `/Settings/Options/Settings/LocaleSettings` content.)

The URL segment, which will be part of the URL that leads to this site, must also be unique. In the unlikely event that the calculated URL segment already exists, the field will be marked red and you have to insert a new value.

If one of the site manager groups has not been created yet, the corresponding field will be marked red. Enter names of existing user groups separated by comma or create one for the calculated name. See also [Section 3.15, “User Administration”](#) in *Content Server Manual*. When you want to use a new group that you have to create first, you have to restart the deriving process after you have created the group.

### Folder Rights

If some of the given groups do not have sufficient permissions to access the newly created site, they will get required rights assigned. New rights apply directly to the newly created site-root-folder. Note, that this does not apply to possibly required rights outside of the site folder, like for example global folders for Themes and Settings.

It is recommended, that the chosen groups have all required workflow roles applied, which especially the group `translation-manager-role`. Role assignment is not part of the site derivation process.



- Click **OK**. The site will now be derived. This can take some time, depending on the size of your site.

## Translation, Localization, Globalization | Preparing Translation: Find Content Items That Need Translation

If the site folder does not exist, yet, it will be created. Adjust its rights rules if desired. For example, the site manager group has no rights on the newly created site folder.

You see the derived site as a subsite of the master site.

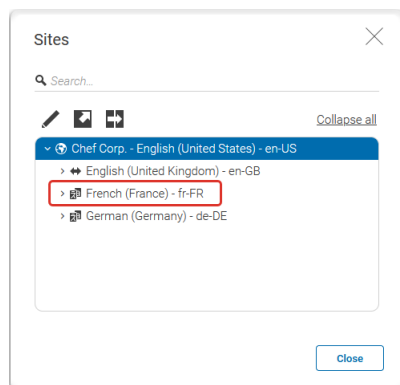


Figure 5.11. Derived Site in Sites Window

### 5.4.2 Preparing Translation: Find Content Items That Need Translation

To support you in finding content items in a site that need to be translated, you can use the Translation State Dashboard Widget and the Translation State Library Filter. The dashboard widget gives you an overview of the translation state of the selected site. It displays five different states:

- *New in Master*

Content items that only exist in the master and not in the local site

- *Master Updated*

Content items that have been localized to the local site but in the meantime have a new version in the master site, and are not in translation, yet

- *Not Translated Yet*

Content items that have been copied to the local site when the site was cloned, but have neither been translated nor are currently in translation

- *In Translation*

# Translation, Localization, Globalization | Preparing Translation: Find Content Items That Need Translation

Content items that are already being translated to the local site in some version

- *Up to Date*

Content items the current versions of which have been translated to the local site

The local site can be configured in the widget's edit mode. For each of the states the number (absolute and relative) of content items in this state are visualized as a bar chart.

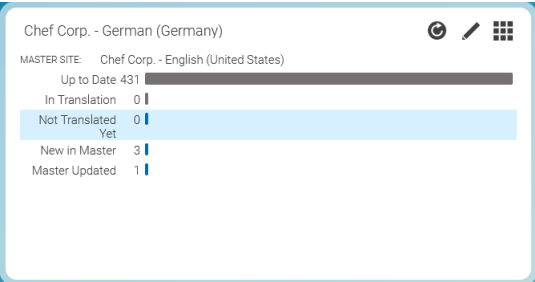


Figure 5.12. Translation State Dashboard Widget

A click on a row of the chart opens the library in search mode with the library filter set to the translation state of the clicked bar. The library then shows the content items in the master site for the selected states, so that you can directly start a translation from here.

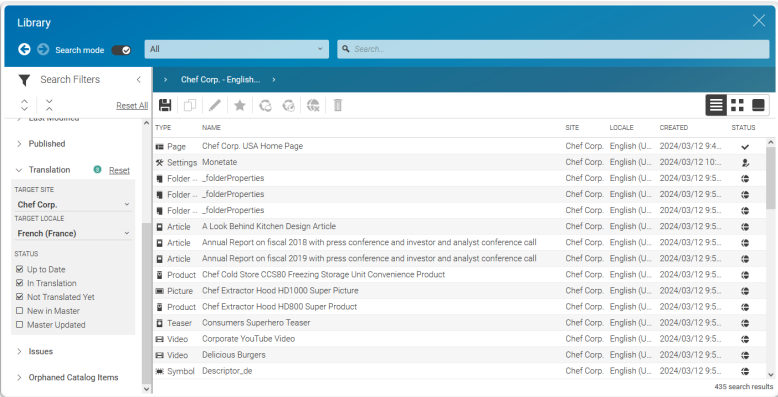


Figure 5.13. Translation State Library Filter

## 5.4.3 Performing Translation: Start, Finish and Abort

You can use translation workflows to coordinate translations in the derived sites. As site manager of a master site, you may trigger workflows to delegate translation to the site managers of the derived sites. As site manager of a derived site, you may trigger a workflow to retrieve updated contents from the master site for translation.

### NOTE

To start a translation workflow the rule applies, that you have to be a member of the group translation-manager-role. In addition to that, you generally need to be either set as a site manager of the master site or as a site manager of the target site (or sites).

More details and alternatives to the rule, that may be relevant for administrators, for example, can be found in [Groups and Rights Administration for Localized Content Management](#) in *Blueprint Developer Manual*.

For more information on administrating users, take a look at [Section 3.15, "User Administration"](#) in *Content Server Manual*.



If content does not exist in a target site, it will be created automatically. This also applies to links in the original content that have not been translated yet. Thus, your project will also contain these contents.

Contents will always be created in a folder structure identical to that of the master content. So if this folder has been removed or renamed in the target site, it will be created again.

### 5.4.3.1 Translation Actions

While the process of starting a translation differs slightly (see [Section 5.4.3.2, "Translation to Derived Sites" \[128\]](#) and [Section 5.4.3.4, "Translation to Preferred Site" \[131\]](#)) the actions you may perform while translating contents are always the same. Once the translation workflow detail panel opens ([Figure 5.14, "Translation workflow detail panel next steps" \[127\]](#)), you have several options:

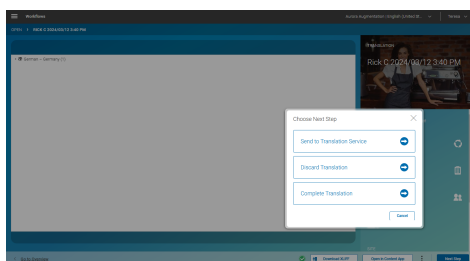


Figure 5.14. Translation workflow detail panel next steps

- You decide that the content needs no translation (see [section “Reject Changes” \[127\]](#))
- You want to translate the content manually in *Studio* (see [section “Manually translate the content” \[127\]](#))
- You want to translate the content with a translation memory system using XLIFF (see [section “Translating the Content with XLIFF” \[127\]](#))

### Reject Changes

The translation of the offered content is not needed for this site. By rejecting the translation, content that has been created automatically by the translation workflow will be deleted.

1. Select the *Discard Translation* link and confirm your choice in the opening dialog with **[Yes, continue]**.

### Manually translate the content

1. Double click the content that you want to translate. It opens in the master side-by-side view. Fields that need translation are highlighted with a green border.
2. Start to translate the content.
3. When you are done with the translation, select *Complete Translation* and confirm your choice in the opening dialog with **[Yes, continue]**.

The translation workflow is finished. The master versions of the translated contents will be set to the current version of their masters automatically.

### Translating the Content with XLIFF

The XML Localisation Interchange File Format (XLIFF) is a standard format for translations with translation memory systems. With *CoreMedia Experience Platform* you can export and import content as XLIFF files.

1. Export the content as an XLIFF file by clicking the button **[Download XLIFF]** in the Workflow App.

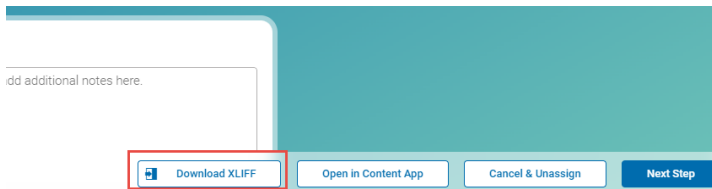


Figure 5.15. Download XLIFF file

The file will be stored in the download folder of your system.

2. Depending on your review process you might now either choose to wait and keep the translation workflow assigned to yourself or you might consider choosing *Send to Translation Service* and click **[Apply]**. In this case a new task, a review task, will be created which is offered to all responsible site managers. Thus, after the results of the translation service are available anyone might take the task and continue the workflow as stated below.
3. When you have finished the translation, import the XLIFF file again as described in [section "Importing an XLIFF file" \[128\]](#).
4. When you have imported the XLIFF file select *Finish Content Localization* and click **[Apply]**.

### Importing an XLIFF file

Importing a translated XLIFF file is quite easy. Simply proceed as follows:

1. Open the upload dialog from the *Upload* menu in the Header bar or click the *Upload files* icon in the Library. The upload dialog opens.
2. Drag the XLIFF file into the upload dialog and press **[OK]**.

The imported content item opens after the import is completed. A pop-up window displays issues that may have occurred during the import.

## 5.4.3.2 Translation to Derived Sites

In order to start a workflow as a manager of the master site to delegate translation to the site managers of the derived sites proceed as follows:

1. Open a translation workflow window.

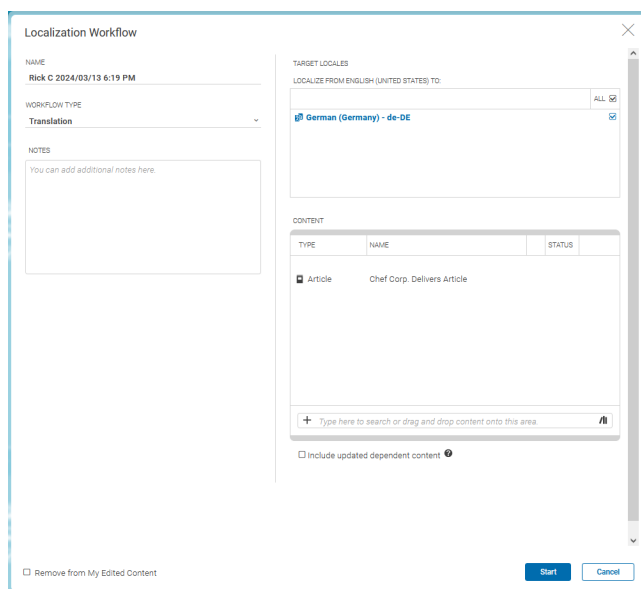


Figure 5.16. Translation Workflow Window

## 2. Edit the workflow. You can do the following:

- Change the name in *Workflow Name*.
- Decide in the area *Dependent Content Excerpt*, if besides the required content also updated dependent contents should be included.

*Required dependent contents* are those, which are referenced by your selected content and which are missing in at least one of the selected target sites. They are required to keep links within the same site (in-site links) consistent.

*Updated dependent contents* are those, which are referenced by your selected content but already exist in all selected target sites. Their translation is out of date in at least one of the selected target sites.

*Special notes on dependent content:*

- *Excerpt only*

The list of dependent content can be huge. That is why only a limited set of contents (100, by default) is shown in Studio. The workflow will take care of expanding the full list of dependent content.

- *No automatic check-in*

In contrast to your selected content, dependent content will not be checked in automatically. This may cause your translation workflow to fail later.

### Make dependent content explicit

To work around the above issues, consider dragging dependent contents to your explicitly selected content set. The excerpt will be filled again, up to a given limit, and the now explicitly selected contents will not only be automatically checked-in, they will also be respected during content-state validation prior to starting the workflow.

You may subsequently add all dependent contents to the selected content, which ensures that all available pre-validation is performed which reduces the chance of the translation workflow to fail.



- Select the locales to which you want to translate in *Target Locales* (if no locale has been selected, the validator shows an error),
- Add notes in the *Notes* area to provide more information to your coworkers concerning the content included in your translation workflow,
- Chose to remove the translated contents from *My Edited Content* by selecting the *Remove from My Edited Content* checkbox. This setting will be remembered in the user preferences.

3. Click **[Start]** to create a translation workflow for every selected locale.

The translation workflow will appear in the *Translation Workflow* panel in the *Control Room* or in the Workflow App of the users who are members of one of the site manager groups of every target site. Each member will be notified about this.

## 5.4.3.3 Accepting Translation Workflows

Each started translation workflow will be offered to the members of one of the site manager groups of the target site. Those users will find the new workflow in the *Translation Workflow* panel of their *Control Room*.



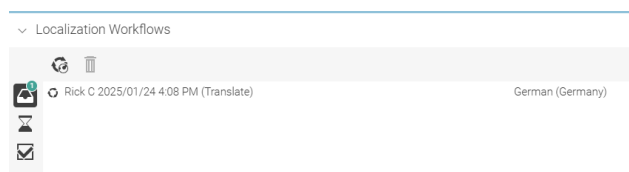


Figure 5.17. Translation workflow panel

1. Double-click the list item in the workflow panel or click the arrow in the selected list item to open the translation workflow in the Workflow App. Check the task, for example by opening content from the workflow with a double-click, and decide whether you want to accept it and perform the translation.
2. Click **[Accept Task]** when you want to perform the translation.

After the task has been accepted, all non-translatable properties in the master content will be applied automatically to the derived content in the translation task. This helps to keep binary and structural data in sync between sites, such as images, crops, settings, and the navigation hierarchy, and complements the XLIFF-based update of translatable properties.

If the merge fails for some reason, a warning will be displayed in the issues of the translation task.

## 5.4.3.4 Translation to Preferred Site

In order to retrieve updates from the master site of your preferred site, you can start a workflow as follows:

1. In the library select contents available in the master site and choose **[Translate into the preferred site]**
2. Click **[Start]** in order to start the workflow
3. In contrast to the process as a global site manager (see [Section 5.4.3.2, "Translation to Derived Sites" \[128\]](#)) the workflow will be directly assigned to you and opened in the *Workflow App*.

## 5.4.3.5 Translation via Drag and Drop

To start a translation you can also drag and drop content into the *Translation Workflows* area in the *Control Room*. The *Localization Workflow* window opens and you can start the workflow.

If the started content belongs to the master site of your preferred site, the workflow will be directly assigned to you and opened in the *Workflow App* as described in [Section 5.4.3.4, “Translation to Preferred Site” \[131\]](#).

## 5.4.3.6 Aborting a Workflow

A translation workflow can only be aborted by the creator of the workflow or by an administrator. You can abort the workflow either in the *Control Room* or in the *Workflow App*.

- Go into the *Translation Workflows* area of the *Control Room*, and select the workflow. Click the *Abort and destroy workflows* icon.



Figure 5.18. Aborting translation workflow in Control Room

- Go into the *Workflow App* and find the workflow in the overview. Right-click on the workflow and select *Abort and destroy* from the context menu.

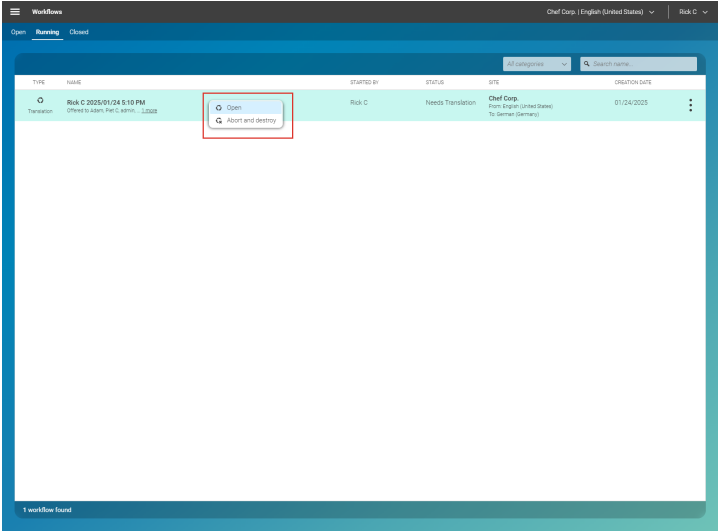


Figure 5.19. Aborting translation workflow in Workflow App

Content items that were automatically created for the translation will *not* be deleted. If you want them to be deleted, you have to reject the workflow instead (see [section “Reject Changes” \[127\]](#)).

# 5.5 Synchronizing Content

*CoreMedia Content Cloud* lets you manage sites in different locales but with the same language. In this use case, it is assumed that most of the content of the derived site should be the same as in the master site. To do so, the content of the derived site can be synchronized with all changes made at the master site. The CoreMedia system supports the synchronization process with the side-by-side view, a synchronization workflow and lets you derive an existing site for synchronization.





Icon	Name
	Open sites window
	Derive site
	Start a localization workflow
	Compare content items

Table 5.2. Icons for Synchronization

NOTE

In addition to the following sections, also have a look into the [Multi-Site Manual](#). The manual describes different options to design your site hierarchy and gives some guidance to avoid common pitfalls when working with multi-site content.



## 5.5.1 Deriving a Synchronized Site

Deriving a synchronized site means that you create a copy of an existing site. The content, structure and appearance of the new site is exactly like that of its master. Afterwards, you can keep the state of the derived site in sync with the state of the master site. A derived site can also be a master site for another derived site.

## Example:

You have a master site, for example an English site for the United States and need another site also in English for Great Britain. The Great Britain site should use all the content from the US site. You also want to have a Canadian site, which uses most of the content of the Great Britain site. So, you derive an Great Britain site from the US Site and the Canadian site from the Great Britain site. When you have changes in the US master site, you start a synchronization workflow. In this case, changes in the master site are propagated to the Great Britain site and also to the Canadian site, by default. This would be continued until no more synchronized sites can be found. However, you can also exclude sites from the synchronization.

In order to derive a site, do the following:

1. In the Main menu select **Sites**. The sites window opens.

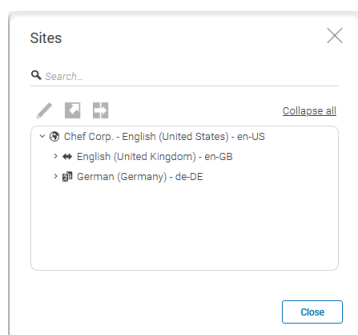


Figure 5.20. Sites Window

2. Select the site you want to derive from and click the *Derive a new Localized Site* icon. Alternatively, you may use the context menu. A modal window opens up.

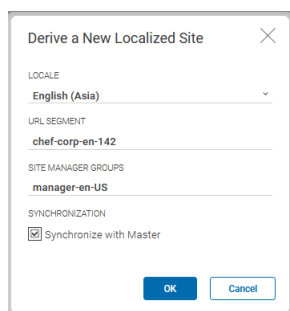


Figure 5.21. Derive a site dialog

3. By default, the *Locale* field will be filled with the locale of the master site and will be marked red, because the locale has to be unique. Therefore, you have to choose a new locale first. Subsequently, new values for the URL segment and the site manager group will be calculated automatically using the selected locale. (You can add new locales to the `/Settings/Options/Settings/LocaleSettings` content.)

The URL segment, which will be part of the URL that leads to this site, must also be unique. In the unlikely event that the calculated URL segment already exists, the field will be marked red and you have to insert a new value.

If one of the site manager groups has not been created yet, the corresponding field will be marked red. Enter names of existing user groups separated by comma or create one for the calculated name (see [Section 8.4, “Creating a New Group” \[212\]](#)). See also [Section 3.15, “User Administration”](#) in *Content Server Manual*. When you want to use a new group that you have to create first, you have to restart the deriving process after you have created the group.

4. Check the *Synchronize with Master* checkbox.
5. Click *OK*. The site will now be derived. This can take some time, depending on the size of your site.

If the site folder does not exist, yet, it will be created. Adjust its rights rules if desired. For example, the site manager group has no rights on the newly created site folder.

You will see now the derived site as a subsite of the master site.

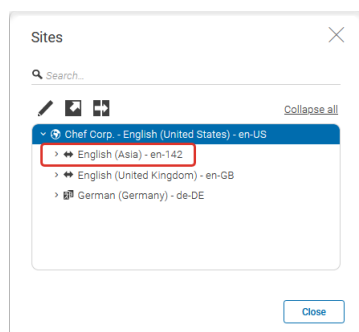


Figure 5.22. Derived synchronized sites

## 5.5.2 Synchronizing Changes between Master and Derived Site

When you have derived a synchronized site, you have to keep it in sync with the master site. To do so, you use the Synchronization workflow.

In order to synchronize the changes you have applied at the master site, do the following:

1. From the master site, select the content items that have changes that you want to synchronize with the derived sites.
2. Open a localization workflow window.

## Translation, Localization, Globalization | Synchronizing Changes between Master and Derived Site

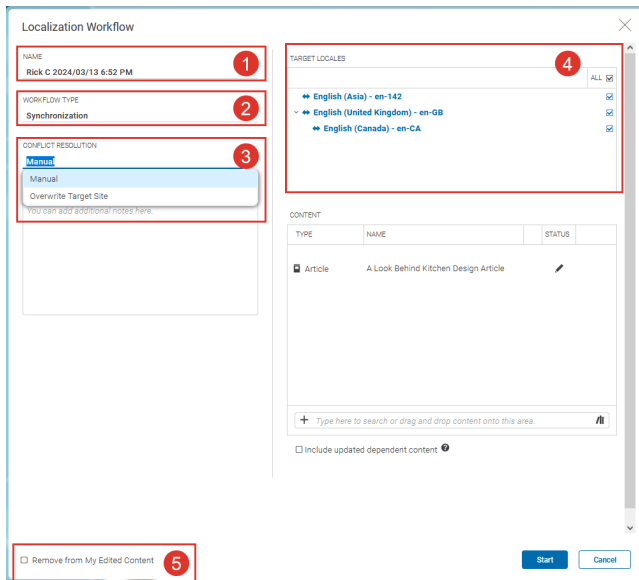


Figure 5.23. Localization Workflow Window

3. In the *Workflow Type* field (2) select the *Synchronization* workflow.
4. Edit the workflow:
  - Content in a synchronized site is not locked or secured in any way. That is, you can edit content of a synchronized site like any other content. However, when both the master content and the content of the synchronized site have changed, then there is a conflict. Select one of the following solution strategies from the *Conflict Resolution Strategy* field (3):
    - Manual

When there are conflicts, the workflow escalates. You have to resolve the conflict and start the synchronization again. You can use the Side-by-side view to compare both versions.
    - Overwrite Target Site

When there are conflicts, the content of the master site will overwrite the conflicting content of the synchronized site.
  - You can change the name in *Workflow* (1),
  - You can choose to remove master contents from *My Edited Content* (5) by selecting the *Remove from My Edited Content* checkbox. This setting will be remembered in the user preferences.



- You can exclude sites from the synchronization in the *Target Locales* field (4). By default, all synchronized sites are selected. When you exclude a site, then all subsites are also excluded and when you include a subsite, then all parent sites are also included.

When you exclude a site that has included subsites, a warning dialog opens up. Click **[Deselect All]** to deselect all subsites. You can disable the warning by checking the *Do not show this message again* checkbox. You can enable the warning again in the *Preferences* dialog.

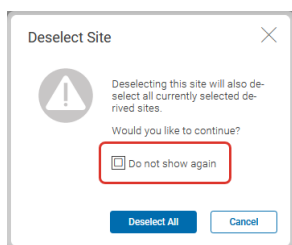


Figure 5.24. Warning for deselected subsites

5. Click **[Start]** to start the synchronization workflow. The workflow runs without further user interaction.

### NOTE

To start a synchronization workflow the rule applies, that you have to be a member of the group translation-manager-role. In addition to that, you need to be set as a site manager of either the current site or any of the sites above within the hierarchy.

More details and alternatives to the rule, that may be relevant for administrators, for example, can be found in [Groups and Rights Administration for Localized Content Management](#) in *Blueprint Developer Manual*.

For more information on administrating users, take a look at [Section 3.15, "User Administration"](#) in *Content Server Manual*.



The changes are applied to all selected synchronized sites, and the changed items appear in the *My Edited Content* field from where you can publish the changes.

**NOTE**

Keep in mind that properties are not synchronized, when you have unchecked the *Keep synchronized with Master* checkbox of the content (see [Section 5.5.3, “Removing Content Permanently from Synchronization” \[140\]](#)) or for which your developers have set the `extensions:automerge` attribute to false in the content type model.



## 5.5.3 Removing Content Permanently from Synchronization

When you have derived a synchronized site, all content of the site is configured to be synchronized with the master site. However, you can remove content items from the synchronization. This is useful when some content items in the synchronized site should be different from the master content.

### Example:

Suppose you have a synchronization site **S** with master site **M** and some synchronized subsites. Removing a content item of **S** from the synchronization cuts it off from updates from the master site **M**. This means, also the subsites of **S** do not receive updates from **M** for this content item. However, they still receive updates of **S** for this content item unless the subsites themselves have removed the content item from the synchronization.

Open the content and, in the *Localization* tab, uncheck the *Keep synchronized with Master* checkbox.

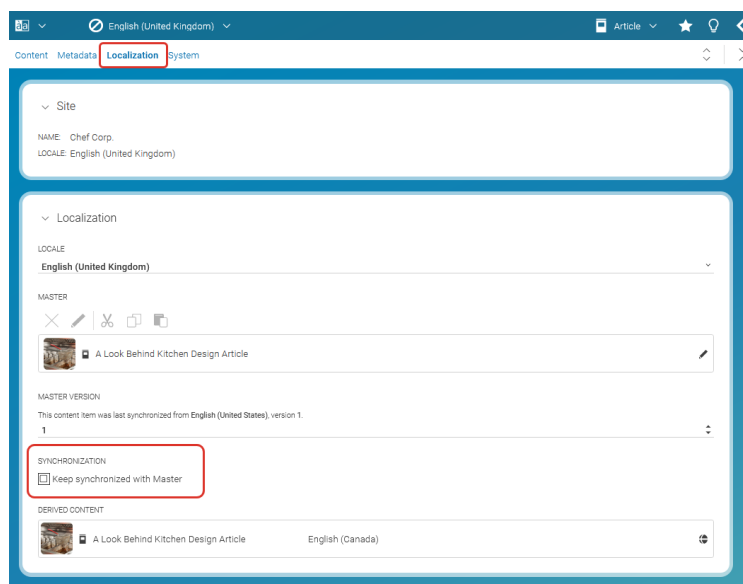


Figure 5.25. Uncheck the synchronization checkbox

From now on, changes in the master content will not be propagated to this content in the synchronized site and to content of synchronized subsites.

## NOTE

Automatically merging can also be deactivated in the content type model by your developer using the `extensions:automerge` attribute. So, there might be properties in your content which will not be automatically merged even when *Keep synchronized with Master* is checked.



## 6. Workflow Management

In this chapter you will find a description of the predefined workflows as well as the workflow actions that are needed to customize existing workflows or define new ones.

# 6.1 Workflow App

The Workflow App is a stand-alone application in which you manage your workflows. You start the workflows from the Control Room in Studio but afterwards, you will manage the workflows in the Workflow App.

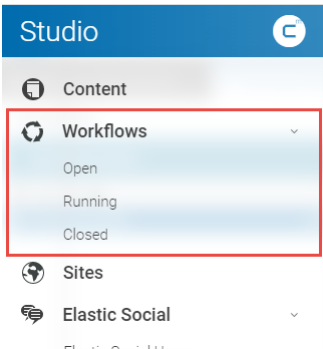


Figure 6.1. Starting Workflow App from the Main Menu of Studio

Figure 6.2, " Workflow app main view " [143] shows the Workflow App in the overview. You see one open workflow with some detail information.

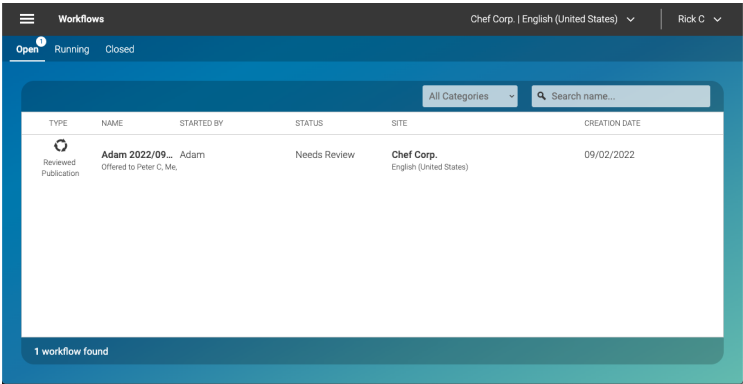


Figure 6.2. Workflow app main view

In this view, you get an overview of all open, running or closed workflows. You can search for workflow names, by using the search field in the top right corner. In order to search for a workflow, you will have to submit your search by pressing

<ENTER> on your keyboard. You can also right-click on a workflow and either select **Open** or **Abort and destroy** from the context menu.

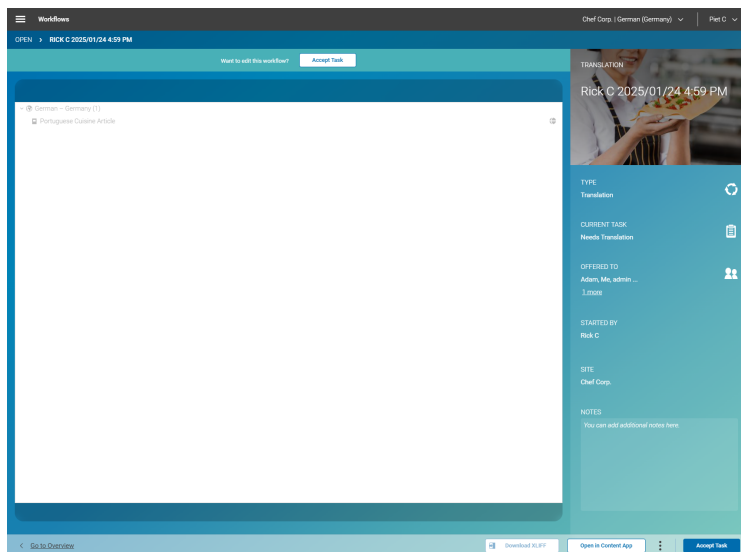


Figure 6.3. Opened workflow in Workflow app

In an opened workflow, you can, for example, open the included content in the Content App of Studio. If you do so, then the Workflow App opens together with the Content App in a split-screen mode.

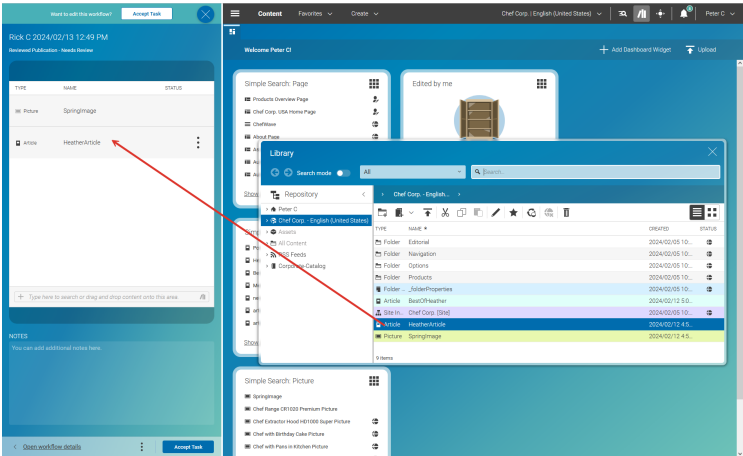


Figure 6.4. Workflow App in split-screen mode with Content App

# 6.2 Publication

In this chapter you will find a description of publication workflows and a description of the publication semantics.

CoreMedia delivers the listed example workflows. But the workflow facilities are not restricted to those features. They can be tailored to fit all types of business processes.

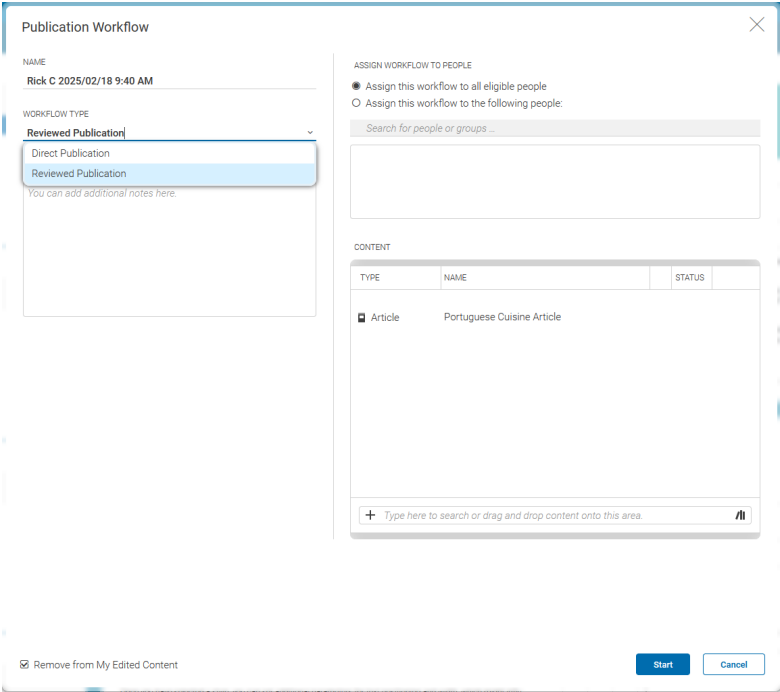


Figure 6.5. Starting a Publication Workflow

## 6.2.1 Approval and Publication of Folders and Content Items

A publication synchronizes the state of the *Live Server* with the state of the *Content Management Server*. All actions such as setting up new versions, deleting,



moving or renaming files, withdrawing content from the live site require a publication to make the changes appear on the *Live Server*.

CoreMedia makes a distinction between the publication of structural and of content changes:

- Content-related changes are changes in content item versions such as a newly inserted image, modified links, text.
- Structure-related changes are moving, renaming, withdrawing or deleting of resources. So it becomes possible to publish structural changes separately from latest and approved content item versions.

For every publication a number of changes is aggregated in a change set. This change set is normally composed in the course of a publication workflow. The administrator and other users with appropriately configured editors can also execute a direct publication, which provides a simpler, although less flexible means of creating a change set.

### Change Set in Direct Publications

When performing a direct publication, the change set is primarily based on the set of currently selected resources or on the single currently viewed resource. As the set of resources does not give enough information for all possible types of changes, three rules apply:

- You cannot publish movements and content changes separately. Whenever applicable, both kinds of changes are included in the change set.
- When a content item is marked for deletion or for withdrawal, new versions of that content item are not published.
- If the specific version to be published is not explicitly selected, the last approved resource version is included in the change set.

There are also some automated extension rules for the change set, which modify the set of to-be-published resources itself. These rules can be configured in detail. Ask your Administrator about the current settings.

- When new or modified content is published and links to an as yet unpublished resource, the unpublished resource is included in the change set. Depending on the configuration, also recursively linked content items can be included in the change set. Target content items that are linked via a weak link property are not included in the change set.
- When the deletion of a folder is published, all directly and indirectly contained resources are included in the change set.
- When the withdrawal of a folder is published, all directly and indirectly contained published resources are included in the change set.

- When the creation, movement, or renaming of a resource in an unpublished parent folder is published, that folder is included in the change set.

### 6.2.2 Predefined Publication Workflows

The predefined workflows for the approval and publication of resources are described in the following.

- Simple publication: Process `StudioSimplePublication` defined in `studio-simple-publication.xml`
- 2-step publication: Process `StudioTwoStepPublication` defined in `studio-two-step-publication.xml`

### 6.2.3 Features of the Publication Workflows

The predefined publication workflows have some features in common, which are described in the following:

#### Users and Groups

In order to execute tasks within workflows, users have to be assigned to special groups. In the predefined publication workflows, these are the following:

1. *composer-role*: to be able to create (and start) a publication workflow and compose a change set
2. *approver-role*: to be able to approve the resources in the change set
3. *publisher-role*: to be able to publish the resources in the change set

Special groups can be defined and linked to the workflow via the `Grant` element in the workflow definition file.

Note that, when all eligible users for a task reject that task, the task is again offered to all eligible users. So if you are the only user for an *approver-role* group and you start a publication workflow, the second step of the workflow will be escalated. That is because you cannot be the composer and the approver of a resource – and there is no other user than you.

#### Basic Steps in a Publication Workflow

After a user has created one or more content items, these content items should be proofread, approved and published in a workflow:

- 1. The user (not necessarily the user who did the editing) starts a workflow. If he selects resources at starting time, these resources will be added to the change set and the compose task will be accepted automatically. Otherwise, he has to add the resources to the change set later.
- 2. The user completes the 'compose' task.
- 3. The task 'approve' is automatically offered to all appropriate users (members of the *approver-role* group, but not to the composer – even if he is a member of this group). Somebody accepts the task and approves the resources.

The user has the following options:

option A	option B	option C	option D
The user accepts the task, approves the resource(s)and finishes the task. All resources are approved.	The user accepts the task, does not approve all resource(s) and finishes the task	The user rejects the task.	The user accepts the task but delegates it to somebody else.
The task 'Publication' is offered to all members of the group <i>publisher-role</i> .	The change set is sent back to the user who completed the 'compose' task.	The task is offered all other members of the group <i>approver-role</i> .	The task is automatically accepted by this user.

Table 6.1. User options.

# 6.3 Translation Workflow

A translation workflow can be used to communicate changes in the project of a master site to the derived sites.

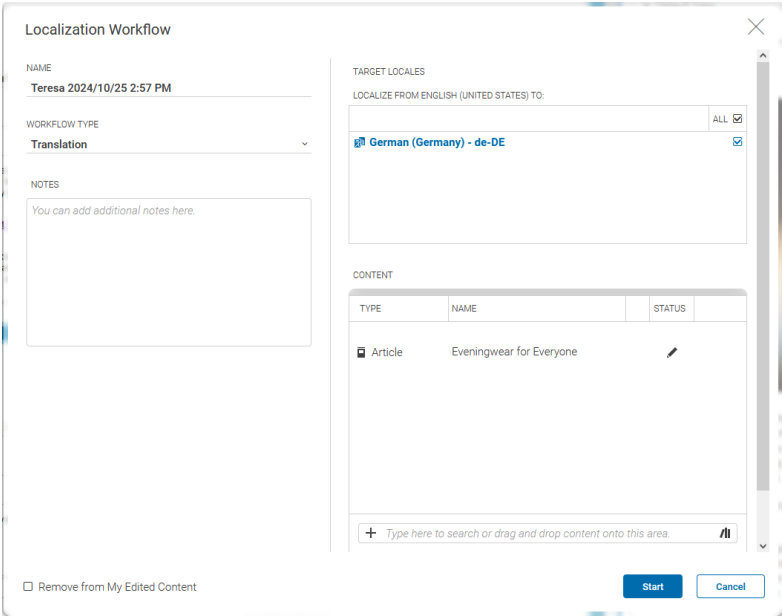


Figure 6.6. Starting a translation workflow

## 6.3.1 Roles and Rights

The translation workflow process is based on two roles defined for *CoreMedia Experience Platform's* Multi-Site concept:

- The group translation-manager-role contains all users that are allowed to start a translation workflow.
- The site manager groups define the users who may accept translation workflows for the content of a site.

## 6.3.2 Configuration and Customization

The example translation workflow is meant to be configured to your needs. You might define multiple translation workflows, like translation via translation agency or manual translation performed by the site managers.

## 7. Basic Content Management

This chapter provides more information on content management basics.

## 7.1 Content Type Model

The basis of the information structure of a CoreMedia system are content types. Content types organize your content and form a hierarchy with inheritance.

*CoreMedia Blueprint* comes with a comprehensive content type model that covers the following topics:

- Common content such as Articles or Pictures.
- Placeholder types that you can use to link to commerce content
- Taxonomies are used to tag content.

### 7.1.1 Common Content Types

#### Requirements

An appealing website does not only contain text content but has also images, videos, audio files or allows you to download other assets such as brochures or software.

In addition, current websites aim to reuse content in different contexts. An article about the Hamburg Cyclclassics might appear in Sports, Hamburg and News section, for example. An image of the St. Michaelis church (the "Hamburger Michel") on the other hand might appear in Articles about sights in Hamburg or religion. Nevertheless, it's not a good idea to copy the article to each section or the image to each article because this is error prone, inefficient and wastes storage.

Therefore, content should be reusable across different contexts (different sites, customer touchpoints for instance) by just applying the context specific layout and without having to duplicate any content. This increases the productivity by reducing redundancy and keeps management effort at a minimum.

#### Solution

*CoreMedia Blueprint* is shipped with content types that model common digital assets such as articles, images, videos or downloads. All these types inherit from a common parent type and can be used interchangeably. In addition, none of these types has fixed information about its context so that it can be used repeatedly and everywhere in your site. The context is first determined through the page which links to the content item or through the position in the folder

hierarchy of the website (see [Section 7.3, “Navigation and Contexts” \[157\]](#) for more details).

## 7.1.3 Content Type Sitemap

### Configuration

The content type Sitemap has three fields you can configure:

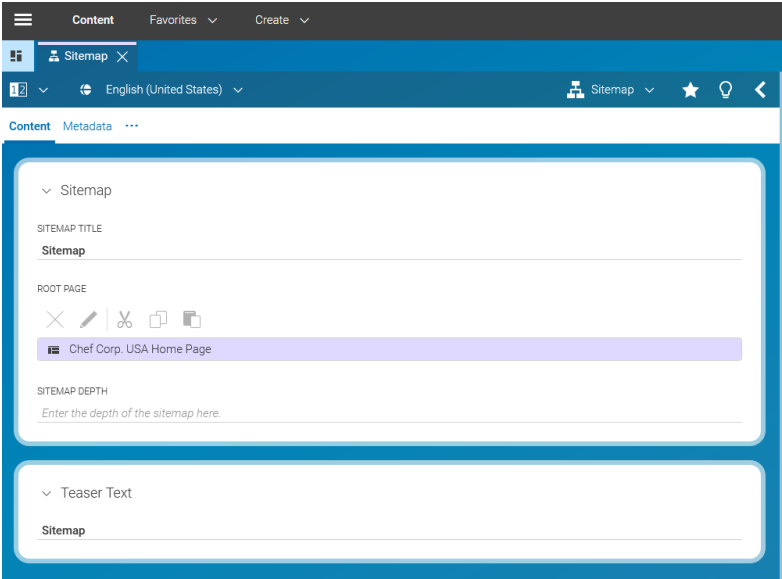


Figure 7.1. Content Type Sitemap

Enter a Sitemap Title which will be rendered as the headline of the Sitemap section in the site. The Root Page field defines the root node from where the content for the Sitemap will be rendered. Additionally, the Sitemap can be rendered to a specific depth which can be set here. This depth is three by default.



## 7.2 Folder and User Rights Concept

It is good practice to organize the content of a content management system in a way that separates different types of content in different locations and to have user groups that attach role depending rights to these locations. This fits with CoreMedia access rights, which are assigned to groups and grant rights to folders and their content, including all sub folders, to all members of that group.

*CoreMedia Blueprint* comes with demo sites that provide a proposal on how to structure content in a folder hierarchy and how to organize user groups for different roles. A more fine grained folder and group configuration can easily be built upon this base.

*CoreMedia Blueprint* distinguishes between the following types of content in the repository:

- **Content:** These are the "real" editorial contents like Articles, Images, Videos, and Products. They are created and edited by editorial users. In a multi-site environment editors are usually working on one of the available sites and they can only access that site's content.
- **Navigation and page structure:** These types represent the site's navigation structure – both the main navigation and the on-page navigation elements like collections or teasers linking to other pages. They are readable by every editorial user, but only the site manager group may maintain them.
- **Technical content types** like options, settings and configuration: These types provide values for drop down boxes in the editorial interface, like view types. They also bundle reusable sets of context settings, for example API keys for external Services. These types are readable by every editorial user but can only be created and edited by Administrators or other technical staff.
- **Client code:** Consists of JavaScript and CSS and is maintained by technical editors.

*CoreMedia Blueprint* comes with a folder structure that simplifies groups and rights management in that way that users taking specific roles only get rights to those contents they are required to view or change. Most notably you will find a `/Sites` folder which contains several sites and several other folders which contain globally used content like global or default settings.

Commonly used content is stored below dedicated folders directly at root level. Web resources like CSS or JavaScript is stored under `/Themes`. Global settings, options for editorial interfaces, and the like are stored under `/Settings`.

### Site-Independent Groups

Along with the site specific groups there are also groups representing roles for global permissions required by some of the predefined workflows. These workflows are especially dedicated to the publication process and are bound to the following roles:

- composer-role

This site-independent group allows members to participate in a workflow as a composer, that is each member of this group may compose a change set for a publication workflow.

- approver-role

This site-independent group allows members to participate in a workflow as an approver, that is each member of this group may perform approval operations within a publication workflow.

- publisher-role

This site-independent group allows members to participate in a workflow as a publisher, that is each member of this group may publish the content items involved in a workflow.

## 7.3 Navigation and Contexts

### Requirements

Websites are structured into different sections. These sections frequently form a tree hierarchy. For example, a news site might have a Sports section with a Basketball subsection. The website of a bank might have different sections for private and institutional investors with the latter having subsections for public and private institutions.

Sections are also often called "navigation" or "context". Usually the sections of a site are displayed as a navigable hierarchy (a "navigation" or "site map"). The current location within the tree is often displayed as a "breadcrumb navigation".

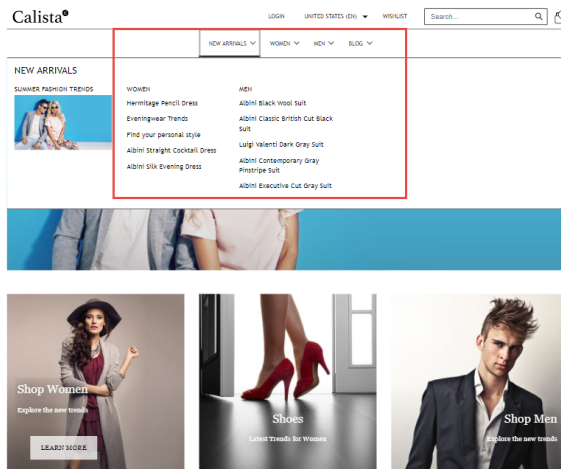


Figure 7.2. Navigation in the Site

Additionally, efficient content management requires reuse of content in different contexts.

For example, reuse of an article for a different section, a mobile site or a micro site should not require inefficient and error-prone copying of that article.

### Solution

A site section (or "navigation" or "context") is represented by a content item of type `CMChannel` or `CMEExternalChannel` which is a child of `CMChannel`. Sections span a tree hierarchy through the child relationships of `CMChannel#children`. If a `CMChannel` is referenced by a `CMSite` item it is considered a root channel, that is an entry into a channel hierarchy representing a website. The `CMChannel` content items fulfill the following purposes:

- **Hierarchy:** They form a hierarchy of site sections which can be displayed as a navigation, sitemap, or bread crumb. Each site consists of exactly one section tree.
- **Context:** They function as contexts for content. Content can be reused within different contexts in different layouts and visual appearance. For example, an article's layout may differ in a company's blog section from its layout in the knowledge base.
- **Page:** Each `CMChannel` can be rendered as an overview page of the section it represents. Therefore, the `CMChannel` contains information about the page structure (the "grid") for this overview page and the pages generated when content items are displayed in the content of the `CMChannel`.
- **Configuration:** `CMChannel` content items contain settings which configure various aspects of the site section they represent. Each `CMChannel` can override parent configuration by defining its own layout settings, content visibility, and other context settings. If for example, the "News" section of a site is configured for post-moderation of comments this configuration can be overwritten to premoderation in the subsection "News/Politics".

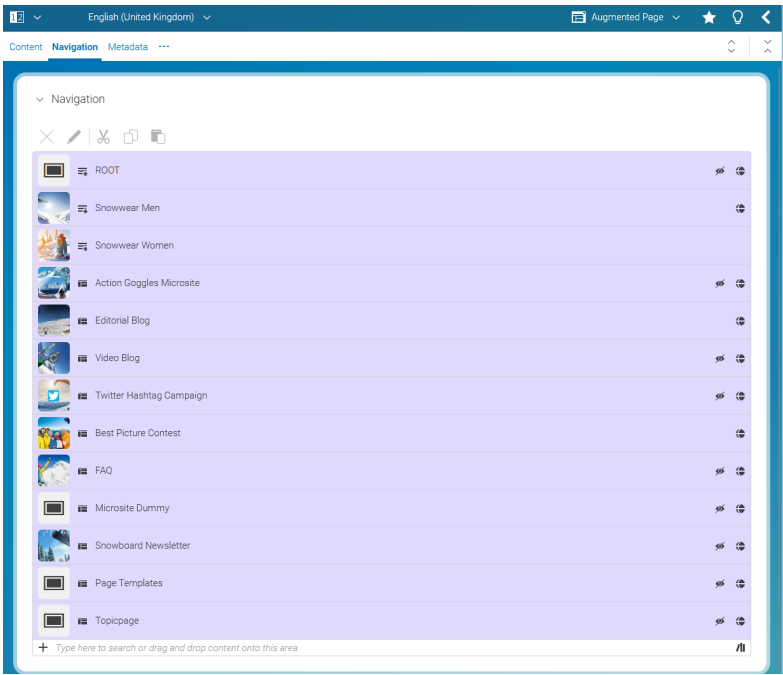


Figure 7.3. Navigation children

## 7.4 Settings

### Requirements

Editorial users must be able to adjust site behavior by editing content without the need to change the code base and redeploy the application. For example:

- Enable/disable comments for a certain section or the whole site.
- Set the number of dynamically determined related content items that are shown in an article detail view.
- Configure the refresh interval for content included from an external live source.

Administrative users must be able to adjust more technical settings through content, for example:

- Manage API keys for external services
- Image rendering settings
- Localization of message bundles

### Solution

*CoreMedia Blueprint* uses Markup properties following the CoreMedia Struct XML grammar to store settings. Struct XML offers flexible ways to conveniently store typed key-value pairs where the keys are Strings and the values can be any of the following: String, Integer, Boolean, Link, Date, Secret, Struct (allows for nested sub Structs).

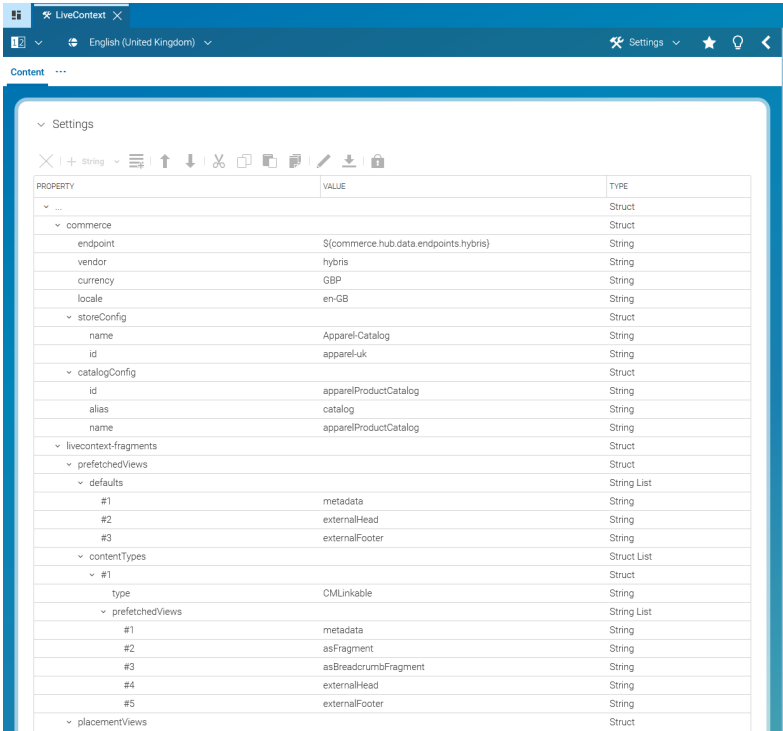


Figure 7.4. Struct editor in a Settings content item

Settings can be defined on all content types inheriting from `CMLinkable`.

The local settings are easiest to edit. However, if you want to share common settings across multiple contents, you should spend the few extra steps to put them into a separate `Settings` content item and add it to the linked settings in order to facilitate maintenance and ensure consistency. Some projects make use of settings quite extensively.

Multiple `Settings` content items are a good instrument to structure settings of different aspects. You can still override single settings in the local settings, which have higher precedence.

The application also considers settings of the content's page context. If you declare a setting in a page, it is effective for all contents rendered in the context of this page.

Settings are inherited down the page hierarchy, so especially settings of the root page are effective for the whole site, unless they are overridden in a subpage or a content.



## 7.5 Page Assembly

### Requirements

#### Requirements

For a good user experience a website should not layout each and every page in a different fancy manner but limit itself to a few carefully designed styles. For example, most pages consist of two columns of ratio 75/25, where the left column shows the main content, and the right column provides some personalized recommendations.

In the best case an editor needs to care only for the content of a page, while the layout and collateral contents are added automatically, determined by the context of the content. However, there will always be some special pages, so the editors must be able to change the layout or the collateral contents. For example for a campaign page which features a new product they may omit the recommendations section and choose a simple one-column layout without any distracting features. In order to preserve an overall design consistency of the site, editors are not supposed to create completely new layouts. They can only choose from a predefined set.

#### Solution

*CoreMedia Blueprint* addresses these requirements with the concept of a page grid and placements.

The page grid does not handle overall common page features such as navigation elements, headers, footers and the like. Those are implemented by Page templates with special views. Neither does the page grid control the layout of collections on overview pages. This is implemented by `CMCollection` templates with special views and view types.

You can think of a page grid as a table which defines the layout of a page with different sections. Each section has a link to a symbol content item which will later be used to associate content with the section. Technically, the layout of a page is defined in form of rows, columns and the ratio between them. A page grid contains no content and can be reused by different pages. So you might define three global page grids from which an editor can select one, for instance.

The content for the page grid on the other hand, is defined in a `CMChannel` content item in so called placements, realized as link lists in structs. Each placement is associated with a specific position of the page grid through a link to a symbol content item. The editor can add content to the placement, collec-

tions for example, which will be shown at the associated position of the page grid.

Placements can also be shared between channels because a child inherits the placements of its parent. A prerequisite for inheritance is that the page grids of the parent and child page must have sections with the same name. For example, the parent channel has a two-column layout with the sections "main" and "sidebar". The child channel has a three-column layout with the sections "main", "sidebar" and "leftcolumn".

For the placements this means:

- The child must fill a placement with content for the "leftcolumn" section, because the parent has no such section.
- The child will override the placement for the "main" section with its content. Inheritance makes no sense for the "main" section.
- The child does not need to declare a "sidebar" placement but can inherit the "sidebar" placement of the parent, even though it uses a different layout.

Before going into the implementation details of the page grid, you will see how to work with page grids in *CoreMedia Studio*.

Editors can manage pages directly by editing the "placements" in the page grid in `CMChannel` content items (localized as `Page` in *CoreMedia Studio*). A placement is a specific area on a page such as the navigation bar, the main column or the right column.

A `CMChannel` can inherit page grid placements of its parent channel. For example, the Sports/Football section of a site can inherit the right column from the Sports section. Editors can also choose to "lock" certain placements and thus prevent subchannels from overwriting them. Each page grid editor provides a combo box to choose between different layouts for a page. Depending on the selected layout, placement may inherit their content if the same placement is defined in the layout of the parent page.

Each placement link list can configure a view type. The view type determines how the placement is rendered.

To define which placement view types are available for a page in some site, view type (`CMViewtype`) content items are placed in view type folders under a site-relative path or at global locations. The default paths are the site-relative path `Options/Viewtypes/` and the absolute path `/Settings/Options/Viewtypes/`. This can be configured via the application property `pagegrid.viewtype.paths`, which contains a comma-separated list of repository paths. Each path may start with a slash ('/') to denote an absolute path or with a folder name to denote a path relative to a site root folder. When

changing these values, please make sure that the existing view type content items are moved or copied to the new target location.

Web pages are represented in the CAE using the `com.coremedia.blueprint.common.contentbeans.Page` object which consists of two elements: the content to be rendered and the context in which to render the content.

Pages where the content to be rendered is the same as the context (for example, section overview) display the page grid of the context. Pages where content items (such as Articles) are displayed within a context use display the context's page grid but replace the "main" placement with the content item.

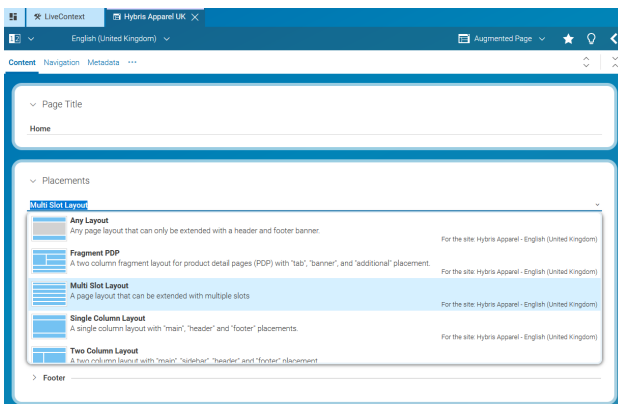


Figure 7.5. The page grid editor

Each placement contains a link list and several additional buttons on top of it. The order of the linked elements can be modified using drag and drop.

The layout of a parent page grid may be changed so that it does not fit anymore with the layout of a child page which inherits some settings. A child may use a three-column layout and inherit most of its content from its parent page that also uses a three-column layout. Then, the layout of the parent may be changed to another layout with a single column that doesn't contain any of the needed layout sections. The child configuration is invalid in this case and the user has to reconfigure all child pages.

```
<Struct xmlns="http://www.coremedia.com/2008/struct"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <IntProperty Name="colCount">2</IntProperty>
  <IntProperty Name="rowCount">1</IntProperty>
  <BooleanProperty Name="disableInheritance">true</BooleanProperty>
  <StructListProperty Name="items">
    <Struct>
      <LinkProperty Name="section" xlink:href="coremedia:///cap/content/550"
LinkType="coremedia:///cap/contenttype/CMSymbol"/>
    </Struct>
  </StructListProperty>
</Struct>
```

```
<IntProperty Name="row">1</IntProperty>
<IntProperty Name="col">1</IntProperty>
<IntProperty Name="height">100</IntProperty>
<IntProperty Name="width">75</IntProperty>
<IntProperty Name="colspan">1</IntProperty>
</Struct>
<LinkProperty Name="section" xlink:href="coremedia:///cap/content/544"
LinkType="coremedia:///cap/contenttype/CMSymbol"/>
<IntProperty Name="row">1</IntProperty>
<IntProperty Name="col">2</IntProperty>
<IntProperty Name="height">100</IntProperty>
<IntProperty Name="width">25</IntProperty>
<IntProperty Name="colspan">1</IntProperty>
</Struct>
</StructListProperty>
<StringProperty Name="name">2-Column Layout (75%, 25%)</StringProperty>
<StringProperty Name="description">Two column layout with main and sidebar
sections</StringProperty>
</Struct>
```

## 7.6 Adding Comments on Content Properties

Editors in *CoreMedia Studio* can create comments on certain content properties. To do so, proceed as follows:

1. Open a content form, with property fields that can be commented.
2. Hovering over a commentable property and click on the button (1) on the right side of the form which appears when hovering.

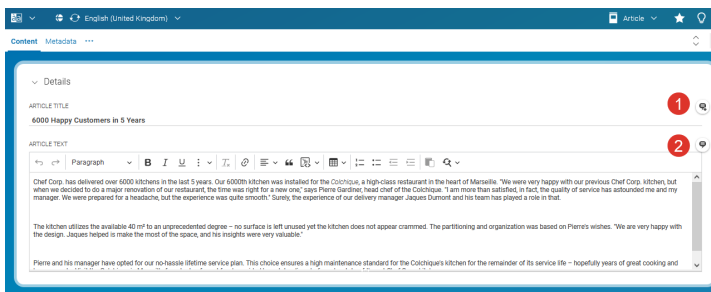


Figure 7.6. Editorial Comments in the Content Form

The comment button will always be visible if comments already exist for that specific property field (2).

3. The Feedback Hub window opens with the *Comments* tab selected. The tab shows all already existing comments for each property field of the content item (1).

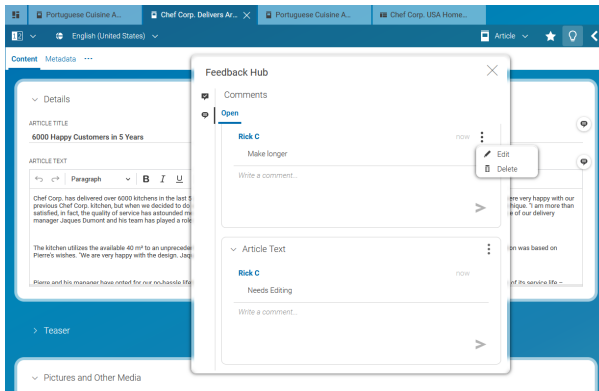


Figure 7.7. Editorial Comments in the Feedback Hub

Write your new comment in the *Write a comment ...* field. If you are ready click the small arrow to send your comment.

### Editing Comments

You can edit your own comments or, as an administrator, edit all comments.

1. Hover over the comment, click the icon (1) with the three dots and select **Edit**.
2. Edit the comment in the *Edit Message* field and click **[Save]** when you are done.

### Deleting Comments

You can delete your own comments or complete threads started by you. As an administrator you can delete all comments and threads.

1. Hover over the comment and click the icon (1) with the three dots or select the icon (2) for the whole field. Select **Delete** from the opening menu.
2. Confirm the deletion in the opening dialog by clicking **[Delete]**.

Users who fulfill one of the following rules will receive a notification when a new comment has been created:

- Users who created a comment within the last 30 days.
- Users who contributed to the content within the last 30 days.
- Users who have the commented content in their edited contents.

# 7.7 Setting a Displayed Date

When you change already published content, you have to publish this change. Of course, the publication changes the publication date of the content. However, you may want that this content always shows the date of the initial publication. To do so, you can set a custom displayed date.

Open the *Content* tab of the content and in the *Displayed Date* field select the option you want to use.

- Publication Date**
- Use the date of the last publication.
- Custom Date**
- Enter the date that should be shown.

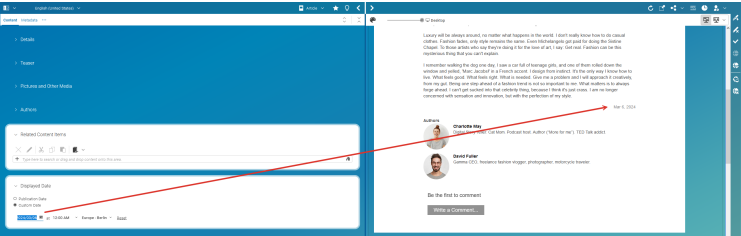


Figure 7.8. Setting a Displayed Date

# 7.8 Time Dependent Visibility

## CoreMedia Blueprint Feature



You can limit the time at which a published content should be visible to the customer. To this end, there are two properties with a slightly different meaning.

- The validity, which defines the period in which a content is allowed to be shown. For example, when you have a time limited license for an image or text. Most of the content items have in the *Content* tab a field *Validity* with the *Valid From* and *Valid To* properties.
- The visibility, which defines the period in which a content should be shown to the customer at a specific position of the website. For example, during a spring campaign. You can select *Visible From* and *Visible To* for content in link lists of placements. For example for pages.

### NOTE

The validity is the leading setting. That means, when the visibility starts before or ends after the corresponding validity dates, then the start or end date of the visibility will be restrained to the validity setting.



## Validity

Validity

VALID FROM

Date at Time Europe - Berlin Reset

VALID TO

Date at Time Europe - Berlin Reset

Figure 7.9. The Validity field

Enter the start time from which the content item should be visible into the *Valid From* properties and the end time into the *Valid To* properties. You can check your settings in the preview and publish the content item as shown below.

When you use the content in a link list, you will see a small icon, which indicates the validity status. (1) means that the content was valid in the past, (2) means that it will be valid in the future, no icon means that the content is currently valid. Hover over the icon to get more information.



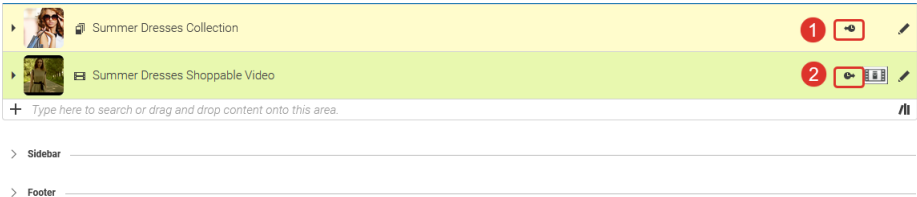


Figure 7.10. Icons for validity

## Visibility

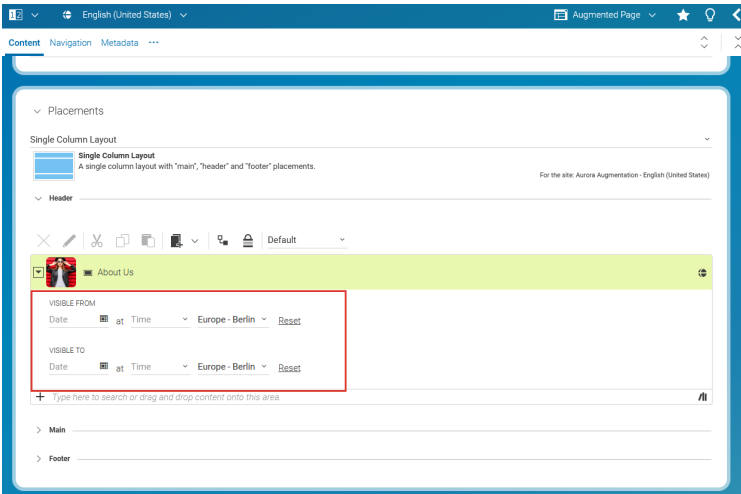


Figure 7.11. The visibility

Enter the start time from which the content item should be visible into the *Visible From* properties and the end time into the *Visible To* properties.

You can check your settings in the preview and publish the content item as shown below.

Check validity and visibility in preview


Icon	Name
	Show preview for a specific time

Table 7.1. Time dependent visibility icon

Check the time settings

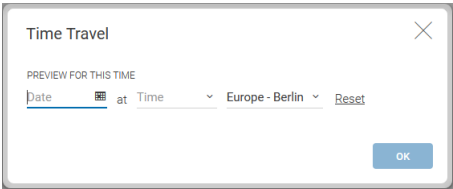


Figure 7.12. Time Travel dialog

You can check your content item with time dependent settings in the Preview. Proceed as follows:

1. Open the content item in the Form and click the **[Show preview for a specific time]** icon. A dialog opens.
2. Enter the time, date and timezone for which you want to check your settings into the dialog. You will immediately see the result of the setting in the preview.
3. When you are satisfied with your result, click **[Reset]** and close the dialog.

# 7.9 Checking Content in External Preview

CoreMedia Blueprint offers a default template set that supports responsive design. That is, your website appears on different devices (PC, Tablet, cellular phone...) in a layout that is appropriate for the device. In order to check the appearance on specific devices, or open the Preview in another browser, you can use the external preview.

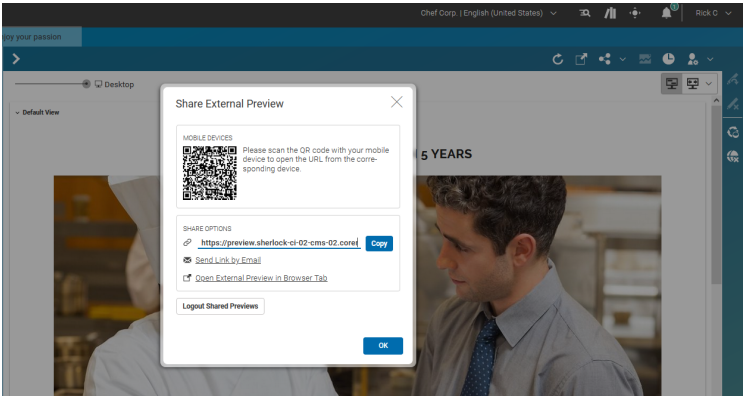



Figure 7.13. Preview dialog

Open external preview

- 1. Click the  icon and select the **Share External Preview** item.

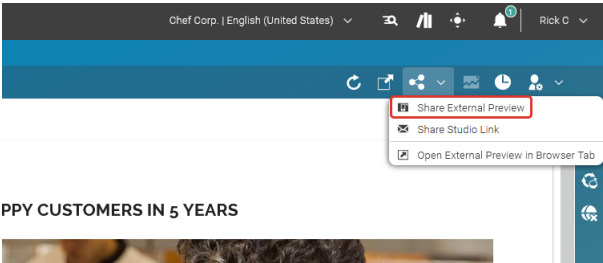


Figure 7.14. External Preview menu

2. A dialog opens up as shown in [Figure 7.13, “Preview dialog” \[173\]](#). Click on **[Open External Preview in Browser Tab]** to preview the current content in a new browser tab. Alternatively, scan the given QR code to open the external preview on mobile devices. You can also copy the link by clicking **[Copy]** or send it by mail by clicking *Send Link by Email*.

Now, the browser or external device shows the preview of the active content item in the Form.

### Logout Connected Previews

You can log out the connected previews by simply clicking the **[Logout Shared Previews]** button and confirming the warning message.

# 7.10 Working with Authors

CoreMedia Studio supports working with authors within an editorial blog. For this, you edit the personal fields (1) and optionally edit additional properties as shown here using the example of an external link (2) to a social media channel of the author. With the help of customized search lists (3), further blog entries are referenced.

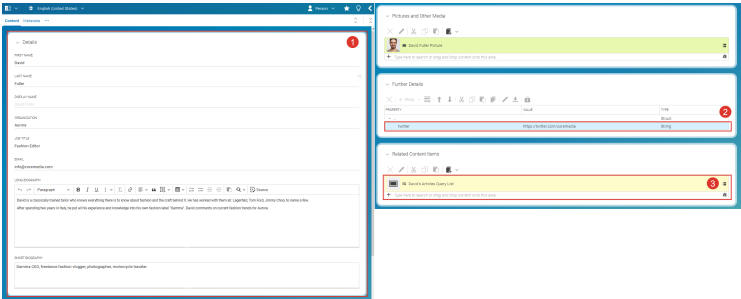


Figure 7.15. Authors

## 7.11 Versioning and Lifecycle

A content item in CoreMedia Studio can only be edited by one user at the same time. In order to work with a content item, the content is first “checked out”. Once checked out, a content item is not available to other users for writing. Each time a content item is checked out, a new, separate version of the content item is created, and the old version is stored for safekeeping. This makes it possible to track changes in the content. Since a new content item version can be created using any previous older version of this content, it is also possible to restore old content.

Before a version of a content item can be published, it must be checked in again and then approved by a user with the appropriate authority to do so. The approval process can be configured: one way is to allow the creator of the content item to approve the it directly. Another method is to let another person — such as the editor-in-chief — take responsibility for content approval (“second pair of eyes” principle). Resources are normally approved and published as part of a workflow process, although it is possible to configure immediate approval and publication.

Links between content item in CoreMedia are generated using unique IDs, meaning that content items can be renamed or moved about, without the link becoming invalidated. However, should a content item queued for publication contains links to content items that have not been published, then its publication would permit “dead” links into the content delivery environment. To prevent this, the CoreMedia Publisher automatically checks to see if publication processes would create such dead links. If the check proves positive, users can automatically include the missing content items to have a sound set not creating dead links.

In CoreMedia, you can compare your current working version of a content item with older versions. This includes highlighting of differences. This is shown in the following figure.

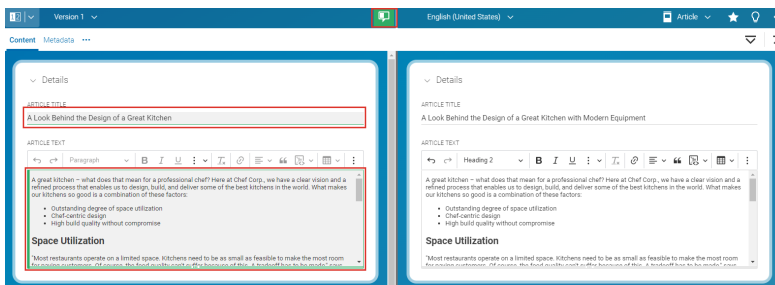


Figure 7.16. Comparing versions

Older versions can be easily restored within CoreMedia Studio.

## 7.12 Validators

To ensure that a content item contains all necessary content, *CoreMedia Studio* supports validators. Validators check if a field fulfills specific conditions. For instance, if a title has been set or a link list contains links to a valid content item. If a condition is not fulfilled, an error or warning is shown, depending on the configuration of your system. You have to correct an error, because otherwise the content item can not be saved or approved (depends on your configuration). You might ignore a warning.

A field with an error will be encircled in red while a field with a warning uses orange. Figure 7.17, “Validators in the Form” [178] shows all GUI elements, used with validators.

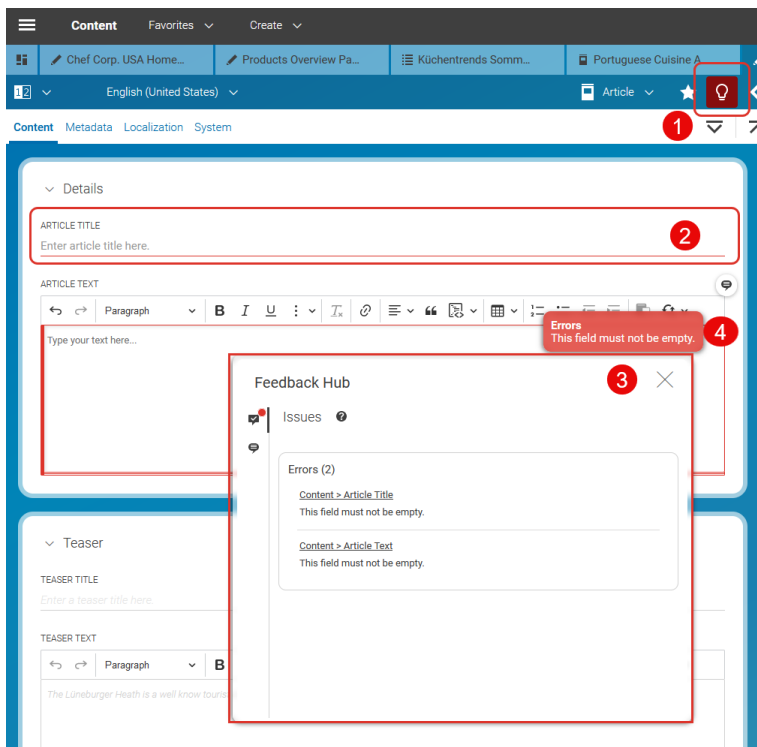



Figure 7.17. Validators in the Form



1. The  icon (1) opens the list with all errors and warnings in the Feedback Hub (3). If there is an error or warning in the current content item, then the icon will be underlined.
2. The faulty field is encircled or underlined in red or orange.
3. The list shows all errors and warnings of the current content item. A click on the underlined text leads you to the faulty field. You can use the list to comfortably complete the mandatory fields of a content item.
4. If you hover with the cursor over the faulty field, a tooltip shows the cause of the error or warning.

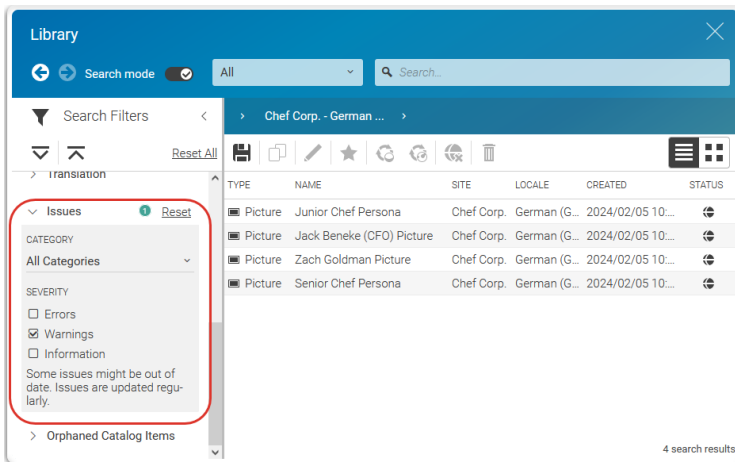


Figure 7.18. Searching for issues in content

You can use the search view of the library to find content items with errors or warnings along with their categories. The "Issues" section of the filter panel makes it possible to filter search results accordingly. Note however, that search results cannot always be up-to-date for all validator types, and that the "Issues" filter section only appears if indexing has been enabled for errors and warnings. The correct state for a content item is always visible in the Form View and in the Feedback Hub.


# 7.13 Getting Keyword Recommendations

You can use the *CoreMedia Feedback Hub* to get feedback for CoreMedia content from an external system. By default, an integration of the Imagga image recognition service (see [imagga.com](https://www.imagga.com)) is provided. The service gives you keywords, describing the image. However, your CoreMedia installation might also be connected to other services, which might offer you more feedback to your content.

In the CoreMedia default websites, the keywords are added to the header of the generated HTML pages.

## Adding keywords to content field

In order to get keywords and add them to a picture content, proceed as follows:

- 1. Open a picture content.
- 2. Click the  icon, to open the Feedback Hub window.
- 3. Select the *Imagga* tab.

You will get keywords in the *Suggested Keywords* field. The percentage shows the reliability of the suggested keyword.

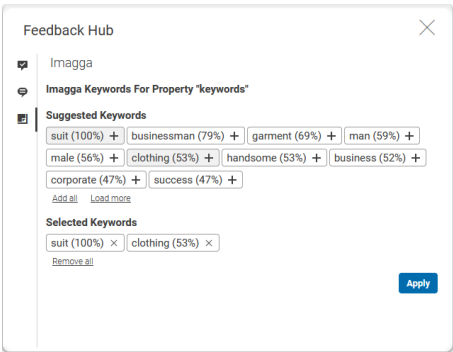


Figure 7.19. Feedback Hub Window with keywords from Imagga system

- 4. Select single keywords by selecting the plus icon or all by selecting **[Add all]**. The selected keywords appear in the *Selected Keywords* field.

5. If you want, you can get more keywords by selecting **[Load more]** , but be aware that all new keywords will have a lower probability to correctly describe the image.
6. Select **[Apply]** to copy the keywords into the configured property of your content.

The selected keywords have been added to your content.

When an error occurs during a feedback request, then the corresponding error message will appear in a red status panel within the Feedback Hub window.

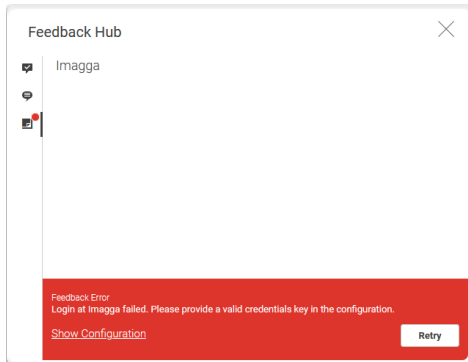


Figure 7.20. Nagbar showing Error during the loading of keywords

# 7.14 Tagging and Taxonomies

## Requirements

Most websites define business rules that require content to be classified into certain categories. Typical examples include use cases such as "Display the latest articles that have been labeled as press releases" or "Promote content tagged with 'Travel' and 'London' to visitors of pages tagged with 'Olympic Games 2012'" etc.

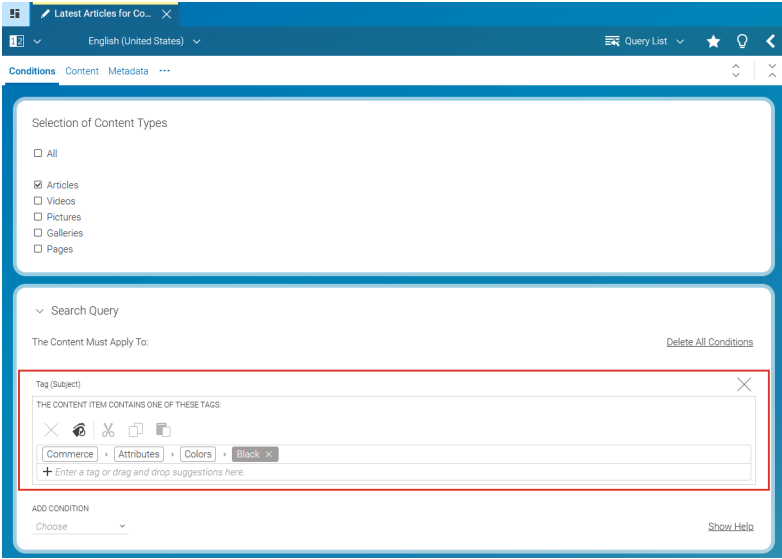


Figure 7.21. Dynamic Query List with taxonomy condition: Black

Keywords or tags are common means to categorize content. Employing a controlled vocabulary of tags can be more efficient than allowing free-form keyword input as it helps to prevent ambiguity when tagging content. Furthermore, a system that supports the convenient management of tags in groups or hierarchies is required for full editorial control of the tags used within a site.

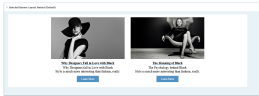


Figure 7.22. Dynamic list of articles tagged with "Black"

### Solution

*Blueprint* currently uses tag information in various ways:

- It is possible to use the taxonomies of a content item as conditions for dynamic lists of content (such as "5 latest articles tagged with 'London'").
- It is possible to display related content for a content item based on content that shares a similar set of tags (see `CMTeasableImpl#getRelatedBySimilarTaxonomies`).

In *CoreMedia Blueprint* tags are represented as `CMTaxonomy` content items which represent a controlled vocabulary that is organized in a tree structure. *CoreMedia Blueprint* defines two controlled vocabularies: Subject and location taxonomies that can be associated with all types inheriting `CMLinkable`.

## 7.14.1 Taxonomy Management

Subject taxonomies can be used to tag content with "flat" information about the content's topic (such as Olympic Games 2012). They can also enrich assets with hierarchical categorization for fine-grained drill down navigation (such as Hardware / Printers / Laser Printer).

Subject Taxonomies are represented by the content type `CMTaxonomy`.

Location taxonomies allow content to be associated with one or more locations. Location taxonomy hierarchies can be used to retrieve content for a larger area even if it is only tagged with a specific element within this area ("All articles for 'USA'" would include articles that are tagged with the taxonomy node North America / USA / Louisiana / New Orleans).

Location taxonomies are represented by the content type `CMLocTaxonomy` which inherits from `CMTaxonomy` and adds geographic information for more convenient editing and visualization of a location.

The taxonomy administration editor can be used to create a taxonomy and build a tree of keywords.

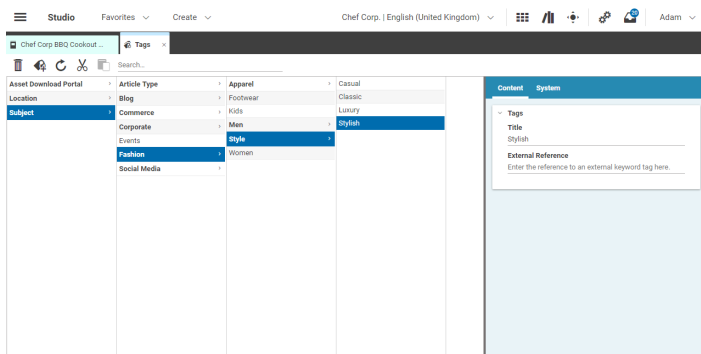


Figure 7.23. Taxonomy Administration Editor

The taxonomy administration editor displays taxonomy trees and provides drag and drop support and the creation and deletion of keywords.

### 7.14.2 Taxonomy Assignment

To enable tagging of content two properties are available the `CMLinkable` content type.

- `subjectTaxonomy`
- `locationTaxonomy`

Editors can assign taxonomies to content items using *CoreMedia Studio* and the Blueprint taxonomy property editor. It allows for the following:

- adding/removing references to taxonomy
- autocompletion
- suggestions

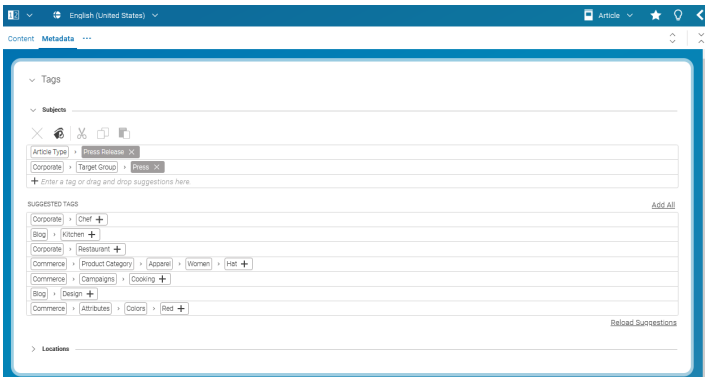


Figure 7.24. Taxonomy Property Editor

The user can add taxonomy keywords to the corresponding property link list using the taxonomy property editor. The editor also provides suggestions that are provided by a name matching algorithm. The strategy type can be configured in the preferences dialog of *CoreMedia Studio*.



Figure 7.25. Taxonomy Studio Settings

To allow a more fine-granular access to taxonomy nodes in the Taxonomy Manager, it is possible to configure roles for different taxonomy trees in the `TaxonomySettings` settings document. The example below shows how to configure the visibility of a global "Subject" taxonomy tree for the user group `taxonomy-admin` and for the site specific "Subject" taxonomy (with site id `"MY_SIDE_ID"`) for the user group `taxonomy-admin`.

Note that this configuration only configures the visibility of the taxonomies inside the Taxonomy Manager. User are still able to see matching contents of the taxonomy in the library if they have the corresponding rights.

If the localization shouldn't be used, the field `defaultLanguage` can be left empty. This will hide the list of localized input fields from the document form. Otherwise, additional `StringProperty` input fields will be shown for every locale of the translations list. The values are stored in the `localSettings` of each taxonomy content item.

ContentSystem

▼ Title (English)

Events

▼ Translations

German (Germany)

Veranstaltungen

Japanese (Japan)

イベント

Figure 7.26. Taxonomy Localization Form

## 7.14.3 Metadata Management

CoreMedia Blueprint feature



*CoreMedia Blueprint* supports categorization of content items by use of taxonomies. The tags attached to a content item can be used to select content items according to their category, in a Content query editor for example. The following screenshot shows a Query List with a taxonomy condition for the Query List editor. Content items are only shown, when they are categorized as "Salad".



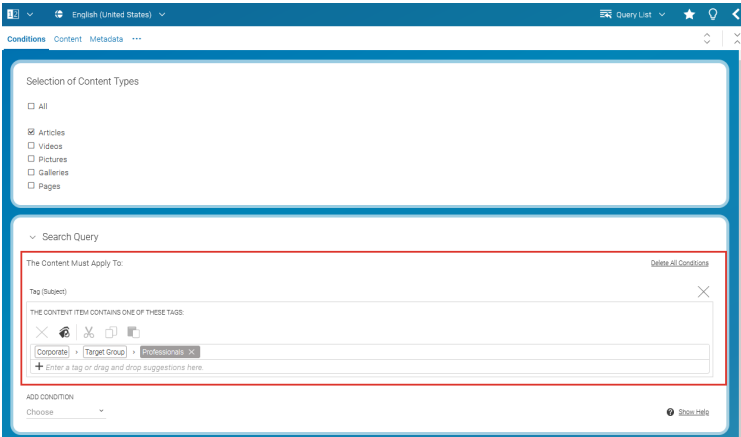


Figure 7.27. Query List with taxonomy condition

CoreMedia Blueprint supports subject taxonomies for information about the content's topic while location taxonomies associate content with a location.



Icon	Name
	Choose Tag
	Add Child Tag

Table 7.2. Taxonomy icons

### 7.14.3.1 Categorizing Content

You can add tags from a predefined taxonomy to a content item.

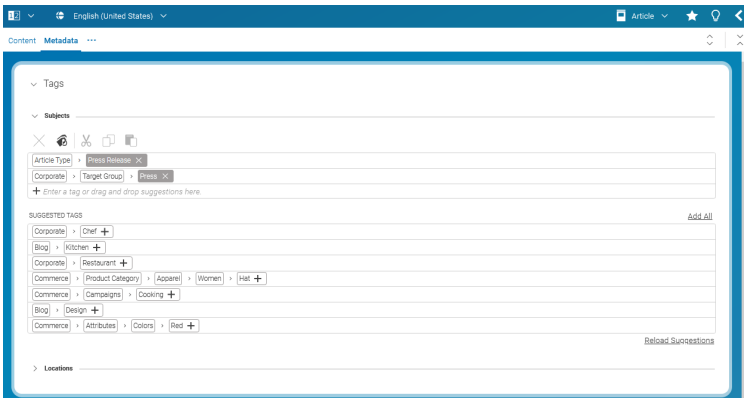


Figure 7.28. Categorized Article

There are three ways:

- Select an entry from the *Suggested Tags* field.
- Start typing. If the word exists in the taxonomy, you will get a suggestion that you can select.
- Click the **[Choose Tag]** icon and select the entry from the Tag Chooser. This step will be explained below.

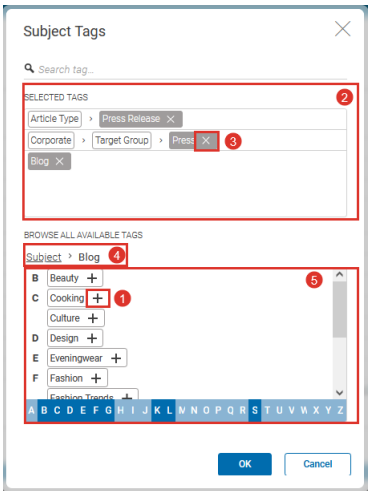


Figure 7.29. Tag chooser

### Using the Tag chooser

All available options are shown in the *Browse all available tags* field (5). There you will find categories (bold face), which have subentries and tags. All selected tags or categories are shown in the *Selected Tags* field (2). The currently selected category is shown at the top of the *Tag* field (4).

#### Adding a tag or category

If you want to add a tag or category to the selected tags, you have to click the + icon of a tag.

#### Removing a tag

If you want to remove a tag or category from the selected tags, you have to click the – icon (3) of a tag or category in the *Selected tags* field (2).

#### Opening a category

If you want to open a category in order to select a sub entry you can click or double-click the category.

### Selecting a suggestions mode

Suggestions can be determined by the system through simple name matching. Additional strategies can be implemented.

*Select a suggestions mode*

1. Open the *User Menu* and select **Preferences**.
2. Open the *Tags* tab and select the mode in the *Suggestions* field.

## 7.14.3.2 Searching for Keywords

The Library has filters for subject and location tags. So you can, for instance, limit your search results to Articles which are tagged with "Monitors".

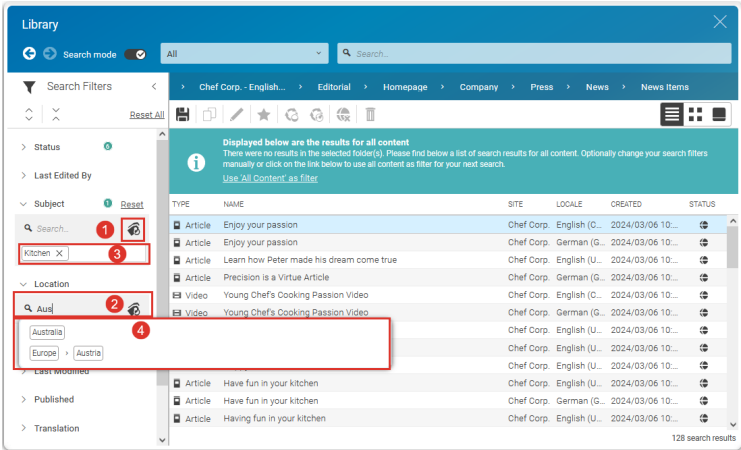


Figure 7.30. Taxonomy filter in the Library

### Adding filtering tags

In order to select filtering tags you can either click the **[Choose Tag]** icon (1) and select the tag from the tag chooser or you start typing in the field (2) and click one of the recommendations (4) from the existing taxonomy.

### Deleting filtering tags

In order to delete a tag, click the – sign (3) beside the tag.

## 7.14.3.3 Managing Taxonomies

A taxonomy can be used to categorize content in a consistent and hierarchical manner. *CoreMedia Blueprint* comes with a predefined taxonomy that you can adapt to your needs using the Taxonomy Editor.

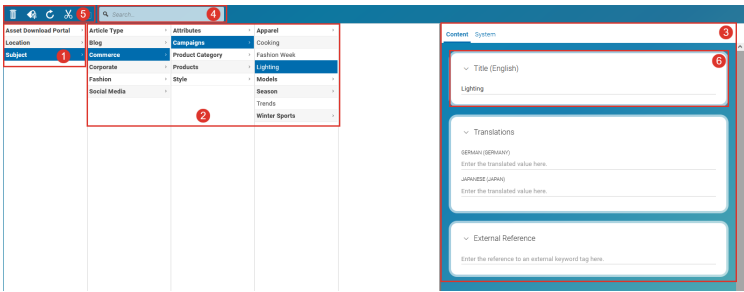


Figure 7.31. The taxonomy editor

Taxonomy entries are stored as content items from type Tag or Location.

### Opening the taxonomy editor

Open the Main menu and click **[Tags]**.

### Editing taxonomies

You can add, move and delete tags in the Taxonomy Editor

#### Adding tags

Select a tag to which you want to add a child tag and click the **[Add child tag]** icon. A form appears where you can enter the name of the new tag (1), translations for the supported languages of Studio (2) and a link to an external reference (3). The newly created tag will automatically be saved and published to the Live System.

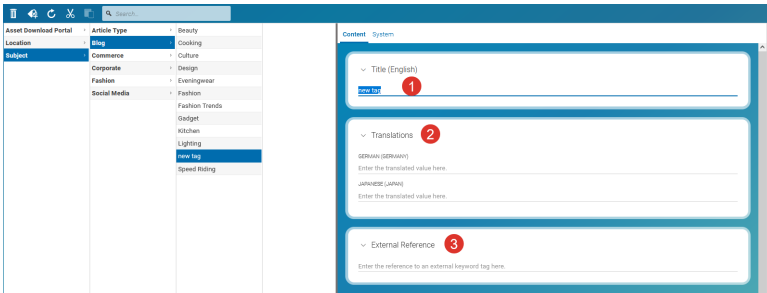


Figure 7.32. Adding a new tag

#### Moving tags

In order to move a tag you can use drag and drop or proceed as follows:

- 192

## 7.15 Content Lists

### Requirements

Websites frequently display content items that share certain characteristics as lists, for example, the top stories of the day, the latest press releases, the best rated articles or the recommended products. Some of these lists are managed editorially while others should be compiled dynamically by business rules defined by editors. It is a common requirement to reuse these content lists across different web pages and use common functionality to place lists on pages and assign different layouts to lists.

### Solution

*CoreMedia Blueprint* defines different content types for lists of content which differ in how they determine the content items. Leveraging CoreMedia's object oriented content modeling these lists can reuse view templates and can be placed interchangeably on web pages.

- CMCollection

A common base type for lists, which all other list types extend. It provides functionality for editorially managed lists.

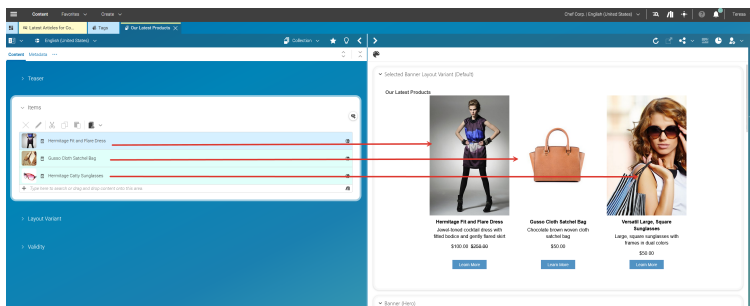


Figure 7.34. A standard collection

- CMGallery

A distinct content type for lists of *CMMedia* content items which should be displayed as a gallery.

- CMQueryList

Dynamic lists that are based on content metadata, such as "latest 5 articles in sport".

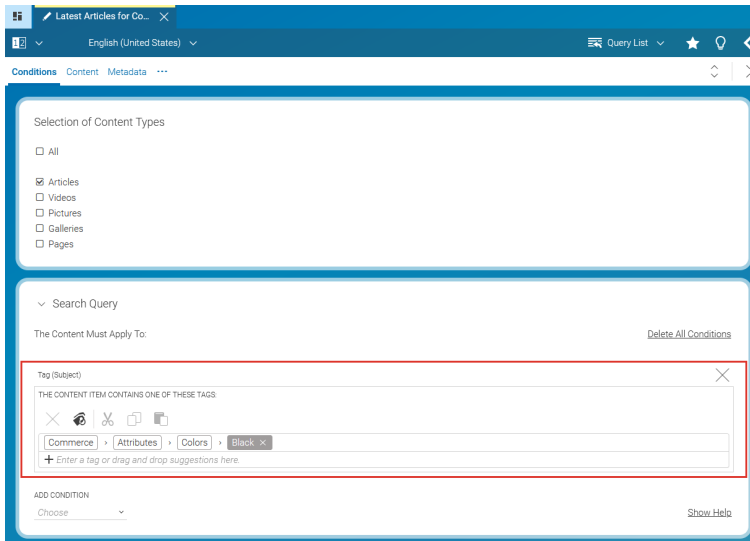


Figure 7.35. Dynamic Query List with taxonomy condition

- ESDynamicList (part of Elastic Social)

Dynamic lists that are based on Elastic Social metadata, such as "5 best rated articles in news."

- Top rated: The N best rated articles
- Most rated: The N most rated articles
- Most commented: The N most commented articles
- Most liked: The N most liked articles
- Most shared: The N most shared articles

You can also determine for which interval the Top-N articles should be taken and how many articles you want to get in maximum.



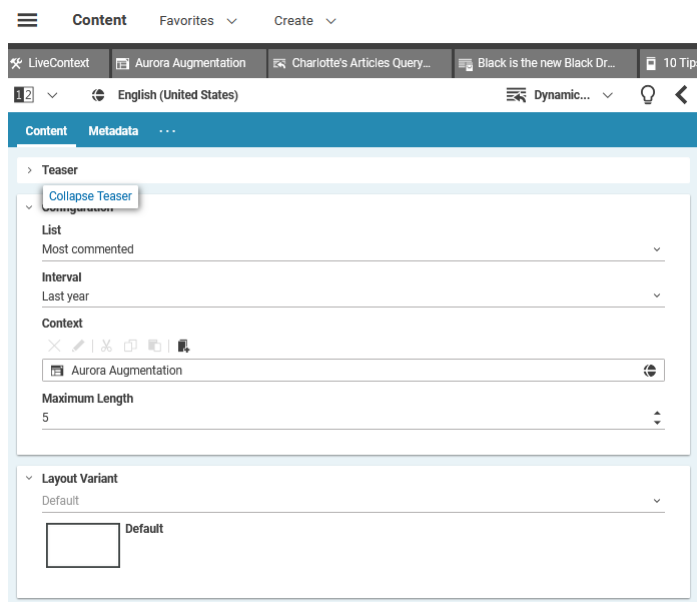


Figure 7.36. Dynamic Elastic Social List

## 7.16 View Types

### Requirements

A common pattern for CoreMedia projects is to reuse content and display the same content item on various pages in different layouts and view variants. A content list, for example, could be rendered as simple bulletin list or as a list of teasers with thumbnails. Similarly, an article can be displayed in a default ("full") view or as a teaser.

Usually the rendering layer decides what view should be applied to a content item in different use cases. For example, the view rendering results of a search on the website could use the `asListItem` view to render the found items.

Editors still need a varying degree of control to influence the visual appearance of content in specific cases. They might want to decide whether a list of content items should be displayed as a teaser list or a collapsible accordion on a page, for example.

### Solution

A dedicated content type called `CMViewtype` is available that can be associated with all `CMLinkable` content types.

### Selecting a view type in CoreMedia Studio

You can use the view type selector which is associated with the view type property to select a specific view type for a content item, a collection for instance. The view type selector is implemented as a combo box providing an icon preview and a description text about the view type. View types can be defined globally or site specific. If the view type item is configured for a site, the name of the site is also displayed in the combo box item.

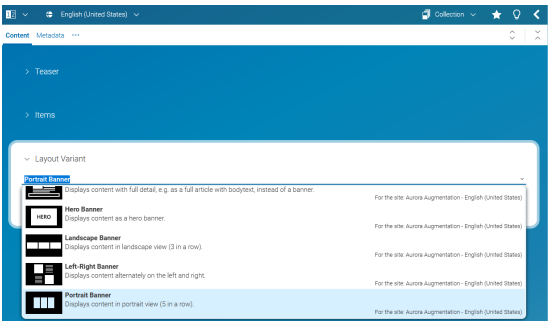


Figure 7.37. Layout Variant selector

## 7.17 Dynamic Templating

### Requirements

In order to quickly implement microsites, campaigns, or specialized channels with unique template requirements, templates can be updated without interrupting the service or requiring a redeployment of the application.

### Solution

Views can be implemented as FreeMarker templates and uploaded to the Content Repository in a container file, preferably a JAR.

## 7.18 Managing End User Interactions

### Requirements

For a truly engaging experience website visitors need to be able to interact with your website. Interactions can reach from basic ways to search content, register and give feedback to enabling user-to-user communication and facilitating business processes such as product registration and customer self care.

End user interactions should be configurable in the editorial interface by non-technical users in the editorial interface of the system. It should, for example, be possible to place interaction components such as Login and Search buttons on pages just like any other content, configure layout and business rules etc.

### Solution

For the Blueprint website, the term "action" denotes a functionality that enables users to interact with the website.

Examples:

- Search: The "search" action lets user to enter a query into a form field. After processing the search, a search result is displayed to the user.
- Login: This action can be used by users to login to the website by adding user name and password credentials. A successful login changes the state web application's state for the user and offers him additional actions such as editing his user profile.

## 7.19 URLs

Link generation and request handling is based on the concepts of the CAE web application. For further information consult the "CAE Application Developer Manual". *CoreMedia Blueprint* offers a simple mechanism for link building and parsing that is based on regular expressions. The out of the box configuration has been made with "SEO Search Engine Optimization" in mind:

- URLs show to which site section the currently displayed page belongs
- URLs for asset detailed pages – opposed to section overview pages – contain the title of the asset

## 7.20 Vanity URLs

Vanity URLs are special human readable URLs which do not contain any technical identifiers like content item IDs. *CoreMedia Blueprint* provides a means to assign vanity URLs to content objects.

Vanity URLs are configured in channel settings. Typically, there is one Vanity URL settings content item for the root channel of a given site. This is the setup chosen for *CoreMedia Blueprint* demo content. To find the Vanity URL settings content item, open the root channel of a site and switch to the **Settings** tab. You will find the Vanity URL settings content item link inside the **Linked Settings** section.

Vanity URLs are defined as a relative URI path. The path might consist of several segments, but if you would like to keep your Vanity URLs simple, just use only one path segment. The URI path is then prepended with a path segment consisting of the site name. For example, for the site `corporate`, a URI path of `my/special/article` would yield the Vanity URL `/corporate/my/special/article`.

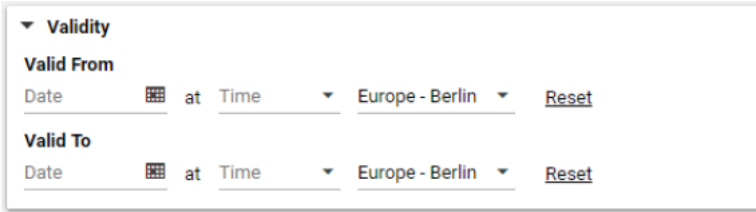
# 7.21 Content Visibility

## Requirements

Content should become available online only within a specific time frame. For example, editors need to ensure that a press release only becomes public at a certain day and time or an article should expire after a specific day. In addition, editors want to preview their preproduced content in the context of the website as if it was already available.

## Solution

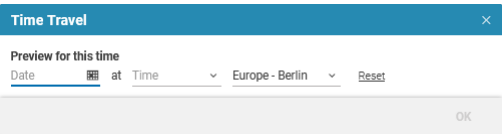
CoreMedia Blueprint supports restricting the visibility of content items by setting the optional `validFrom` and `validTo` date properties of content of type `CMLinkable`.



The image shows a configuration panel for the 'Validity' field. It has a dropdown arrow and the label 'Validity'. Below it are two sections: 'Valid From' and 'Valid To'. Each section contains a 'Date' field with a calendar icon, the text 'at', a 'Time' field with a dropdown arrow, a location dropdown menu currently showing 'Europe - Berlin', and a 'Reset' link.

Figure 7.38. The Validity field

In order to preview the experience at certain points in time, you can use the time travel feature within CoreMedia Studio:



The image shows a 'Time Travel' dialog box with a blue header and a close button. Below the header, it says 'Preview for this time'. There is a 'Date' field with a calendar icon, the text 'at', a 'Time' field with a dropdown arrow, a location dropdown menu showing 'Europe - Berlin', and a 'Reset' link. At the bottom right of the dialog is an 'OK' button.

Figure 7.39. TimeTravelDialog



# 7.22 Robots File

## Requirements

Technical editors should be able to adjust site behavior regarding robots (also known as crawlers or spiders) from search engines like Google. For example:

- Enable/disable crawling of certain pages including their sub pages.
- Enable/disable crawling of certain single content items.
- Specify certain bots to crawl different sections of the site.

To support this functionality most robots follow the rules of `robots.txt` files like explained here: <http://www.robotstxt.org/>.

For example, the site "Corporate" is accessible as `http://corporate.blueprint.coremedia.com`. For all content of this site the robots will look for a file called `robots.txt` by performing an HTTP GET request to `http://corporate.blueprint.coremedia.com/robots.txt`.

## Solution

The so called RobotsHandler will be responsible for requests like this due to the path element `/robots`. The last path element of this URL (in this example `/corporate`) will be evaluated by RobotsHandler to determine the root page that has been requested.

In this example "corporate" is the URL segment of the Corporate Root Page.

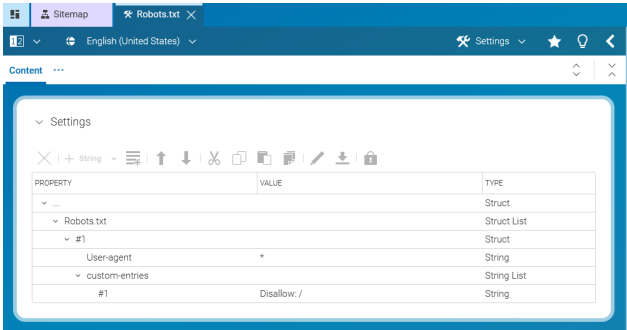


Figure 7.40. Channel settings with configuration for `Robots.txt` as a linked setting on a root page

## 7.23 Sitemap

### Requirements

If you run a public website, you want to get listed by search engines and therefore give web crawlers hints about the pages they should crawl. <https://www.sitemaps.org/> declares an XML format for such sitemaps which is supported by many search engines, especially from Google and Microsoft.

"Sitemap" in terms of <https://www.sitemaps.org/> is not to be mistaken with a human readable sitemap which visualizes the structure of a website. It is rather a complete index of all pages of a site.

The size of a sitemap is limited to 50,000 URLs. Larger sites must be split into several sitemap files and a sitemap index file which aggregates the sitemap files.

### Solution

A sitemap consists of multiple entities (the index and the sitemap files) and has dependencies on almost the whole repository. If a new content is created, which "coincidentally" occurs in the first sitemap file, the entries of all subsequent sitemap files are shifted.

In border cases even the number of sitemap files may change, which affects the sitemap index file. So you cannot generate single sitemap entities on crawler demand, asynchronously and independent of each other, but you must generate a complete sitemap which represents a snapshot of the repository. Moreover, the exhaustive dependencies make sitemaps practically uncacheable, and the generation is expensive. For these reasons *Blueprint* does not render sitemaps on demand but pregenerates them periodically. So you must distinguish between sitemap generation and sitemap service. Both are handled by the live web application, though.

*CoreMedia Blueprint* features separated sitemaps for each site. Sitemap generation depends on some site specific configuration, like the content types to include or paths to exclude, amongst others. This configuration is specified by `SitemapSetup` Spring beans.

## 7.24 Website Search

### Requirements

In order to make content more accessible for their audience virtually all websites have full-text search capabilities. To improve the search experience some websites also offer features such as search term autocompletion, suggestions in case of misspelled search terms, more advanced filtering options or even metadata based drilldown navigation in search results.

### Solution

CoreMedia CMS has built-in integration with the Apache Solr search engine. *Blueprint* comes with a small abstraction layer that offers unified search access to Solr for all CAE based code. It provides the following features, all based on standard Solr functionality:

- Full text search: Search for content across all fields
- Field based filters: Filter results by metadata such as the content type, the site section it belongs to, etc.
- Facets: Display facets, that is the number of results in a field for certain values
- Spellcheck suggestion: "Did you mean" suggestions for misspelled terms
- Search term highlighting: All words are highlighted in your text
- Validity range filtering: Automatically filter for only visible results (see section [Section 7.21, "Content Visibility" \[202\]](#))
- Filter non-searchable: Automatically filter content that should not be part of search results.
- Caching: Search results can be optionally cached for a certain amount of time.

## 8. Managing Users and Groups

### NOTE

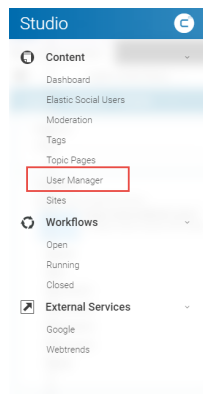
The User Manager is only available for members of the administrator group.



The User Manager is a Studio app that you can use to manage users, groups and rights.

## 8.1 Opening the User Manager

Open the **Main** menu and select **User Manager**.



*Figure 8.1. Open User Manager*

# 8.2 Searching for Users and Groups

1. Open the **Main** menu and select **User Manager**.

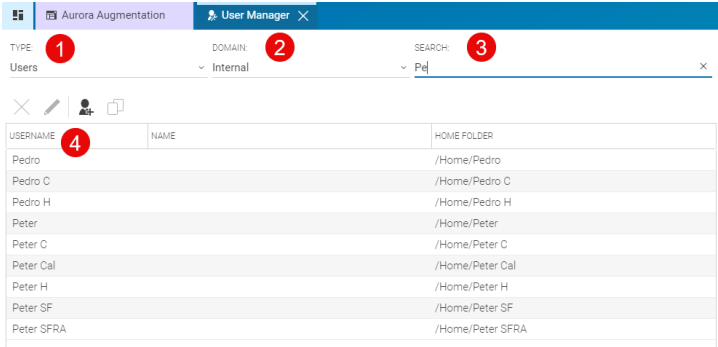


Figure 8.2. Search fields

- 2. In field *Type* (1) select if you want to search for users or groups.
- 3. In field *Domain* (2) select the domain.
- 4. In field *Search* (3) enter your search. The results matching your search are displayed in the list below (4).

**NOTE**

By default, you have to enter at least three characters, to get results. This number can be changed by a system administrator in the property `studio.usermanager.minSearchCharacters`.



- 5. In order to open the detail view, you can either double-click a user or group or select it and click the *Edit* icon.

# 8.3 Creating a New User

### Prerequisite

You have to be a member of the administration group in order to create or edit groups.



In order to create a new user, proceed as follows:

- 1. Open the User Manager.
- 2. Click the **[Create a new User]** button (1).

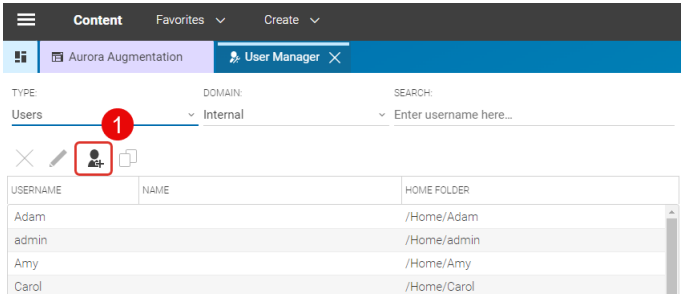


Figure 8.3. Create new user with icon

A window pops up.

- 3. Enter the Username, the password for login and the email address in the corresponding fields and the display name in the *Name* field.

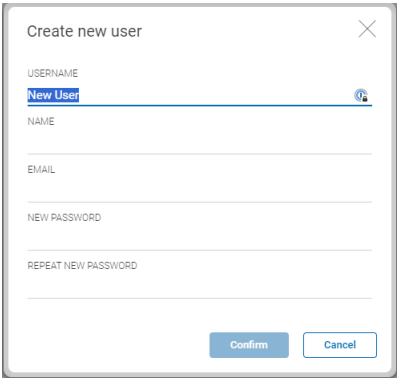
A modal window titled "Create new user" with a close button in the top right. It contains four text input fields: "USERNAME" (with "New User" entered), "NAME", "EMAIL", and "NEW PASSWORD". Below the "NEW PASSWORD" field is a "REPEAT NEW PASSWORD" field. At the bottom right are "Confirm" and "Cancel" buttons.

Figure 8.4. Create new user popup windows

4. Click **[Confirm]** to start the creation. The detail view of the user opens up.

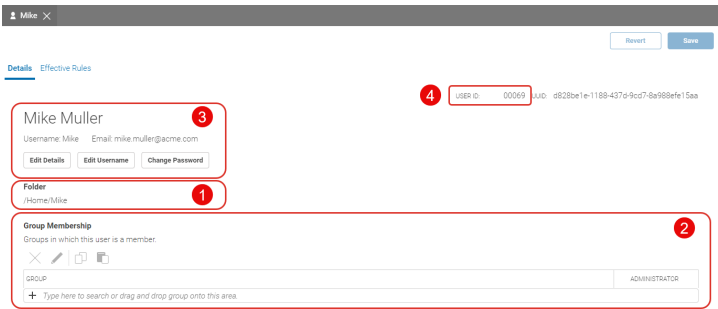
A user detail view for "Mike Muller". At the top is a header bar with "Mike" and a close button, and "Revert" and "Save" buttons on the right. Below is a "Details" tab. The user's name "Mike Muller" is at the top left, with a red circle 3 next to it. Below the name are fields for "Username: Mike" and "Email: mike.muller@acme.com", with buttons for "Edit Details", "Edit Username", and "Change Password". Below this is a "Folder" section with a red circle 1 next to it, showing the path "/Home/Mike". At the bottom is a "Group Membership" section with a red circle 2 next to it, showing "Groups in which this user is a member." and a list of groups including "ADMINISTRATOR". A red circle 4 points to the "USER ID" field at the top right, which contains "00069".

Figure 8.5. Set home folder and group membership

The *Folder* field (1) is showing the default home folder of the user. You can edit the name and email by clicking *Edit Details*, the username by clicking *Edit Username* and change the password by clicking *Change Password*. When you change the username, you also have to enter a new password. The name of the Home Folder of the user will not be changed.

A home folder stores personal content items, as well as queries or settings.

**NOTE**

By default, a user has no rights on its home folder. You have to add the user to a group which has appropriate rights on this folder. See, for example, the predefined staff group.





- 5. In order to be able to do anything, the user needs membership in one or more groups. Click in the link list of the *Group Membership* field (2) and select groups from the dropdown list.
- 6. If you want, you can check the effective rules of the user by opening the *Effective Rules* tab. You will see all the rights of the user on folders and content types due to the membership of the user in groups.

global-site-manager < global-manager <

Revert

Save

Details Rules Effective Rules

PATH	TYPE	READ	EDIT	DELETE	APPROVE	PUBLISH	SUPERVISE
/Assets	Asset	✓					
/Assets	Folder	✓					
/Home	Content	✓	✓	✓			✓
/Home	Editor Preferences	✓	✓				
/Home	Editor Profile	✓					
/Home	Folder	✓	✓				✓
/Settings	Content	✓					
/Settings	Folder	✓					
/Settings/Meta/Mail	Content	✓					
/Settings/Meta/Mail	Mail	✓	✓	✓	✓	✓	
/Settings/Meta/Mail	Folder	✓					
/Settings/Options/Bundles	Content	✓					
/Settings/Options/Bundles	Settings	✓	✓	✓	✓	✓	
/Settings/Options/Bundles	Folder	✓					
/Settings/Options/Settings	Content	✓					
/Settings/Options/Settings	Settings	✓			✓	✓	
/Settings/Options/Settings	Folder	✓					
/Settings/Taxonomies	Content	✓					
/Settings/Taxonomies	Tag	✓	✓	✓	✓	✓	
/Settings/Taxonomies	Folder	✓					
/Themes	CoreMedia Blueprint Object	✓	✓	✓	✓	✓	
/Themes	Template Set	✓		✓	✓	✓	
/Themes	Folder	✓	✓		✓	✓	

Figure 8.6. Check effective rules

- 7. Click **[Save]** in order to save the newly created user.
- If you want to discard changes, click **[Revert]**.

# 8.4 Creating a New Group



### Prerequisite

You have to be a member of the administration group in order to create or edit groups.

A group contains rules on CoreMedia resources. In order to create a new group, proceed as follows:

- 1. Open the User Manager.
- 2. Click the **[Create a new Group]** button (1).

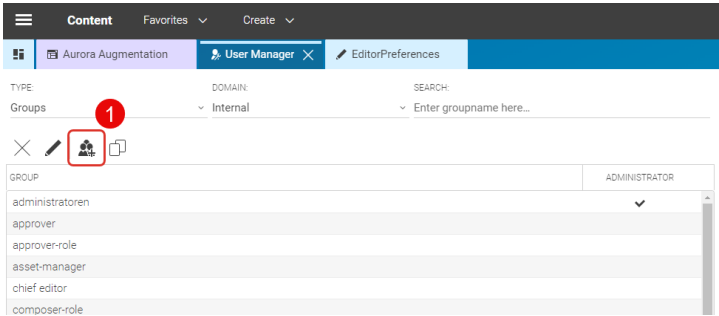
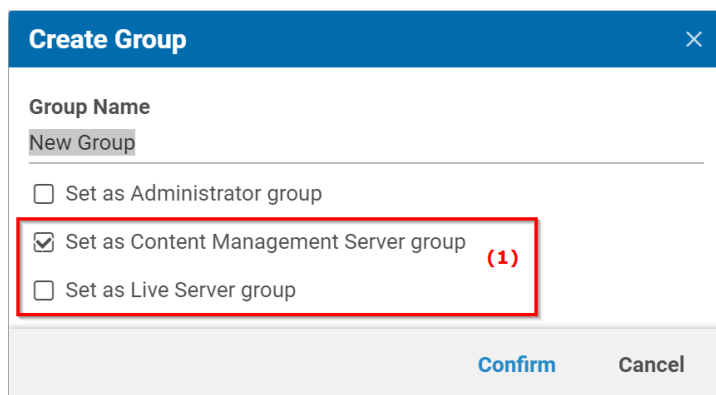


Figure 8.7. Create new group

A window pops up.

- 3. Enter the group name in the corresponding field. If the group should have administrator rights, check the corresponding checkbox (see [Section 3.15.3, “Administrator Groups”](#) in *Content Server Manual* for details on the administrator group).



**Create Group** ×

**Group Name**  
New Group

☐ Set as Administrator group

☒ Set as Content Management Server group (1)

☐ Set as Live Server group

Confirm Cancel

Figure 8.8. Create a new group popup windows

4. Your administrators might have configured the use of Live Server groups (a deprecated feature). In this case, you will see the two options in field (1). Live Server groups can be used to manage access rights to content on the live side of *CoreMedia Experience Platform* (see [Section 3.15.5, “Live Server Groups and Users”](#) in *Content Server Manual* for more details on live server groups). *Content Management Server group* is the default setting, which is also used when no Live Server groups are activated. Select the checkbox of the group type you want to use.

Click **[Confirm]** to start the creation. The detail view of the group opens up.

The screenshot displays the 'Details' tab for a group named 'LiveServerGroupPremium'. The interface includes the following elements:

- Group Name:** A text field containing 'LiveServerGroupPremium' with an 'Edit Group Name' button below it. (1)
- Group ID:** A field showing 'Group ID: 00915'. (3)
- Group Type:** A section with three checkboxes: 'Set as Administrator group' (unchecked), 'Set as Content Management Server group' (checked), and 'Set as Live Server group' (checked). (2) and (4)
- Users:** A section titled 'Users assigned to this groups are visible here.' with a search bar and a table with columns 'Username', 'Name', and 'Home Folder'. It shows 'No assigned users found.' (5)
- Group Membership:** A section titled 'Groups in which this group is a member.' with a search bar and a table with columns 'Group' and 'Administrator'. It shows 'No assigned groups found.' (6)
- Subgroups:** A section titled 'Subgroups Groups that are members of this group.' with a search bar and a table with columns 'Group' and 'Administrator'. It shows 'No assigned groups found.' (7)

Figure 8.9. Set group members and membership

If you made errors during the creation of the group, then you can edit the name (1) and add or remove the administration rights (2). You can also see the ID of the group (3). The group type in field (4) is only shown when Live Server groups are activated. They can not be changed in this window.

5. A group can be a member of another group, where a subgroup inherits the rules of its parent group. In addition, a group can have assigned users.

In order to add groups and users, proceed as follows:

- a. Click in the link list *Group Membership* (6) and select groups from the dropdown list from which the new group should be a subgroup.
  - b. Click in the link list *Subgroups* (7) and select groups from the dropdown list which should be subgroups of the new group.
  - c. Click in the link list *Users* (5) and select users from the dropdown list which should be members of the new group.
6. Open the *Rules* tab and add new rules. See [Section 8.5, "Adding, Deleting and Editing Rules"](#) [216] for adding rules.
  7. If you want, you can check the effective rules of the group by opening the *Effective Rules* tab. You will see all the rights of the group on folders and content types due to the defined rules and the membership in other groups.

RevertSave

DetailsRulesEffective Rules

PATH	TYPE	READ	EDIT	DELETE	APPROVE	PUBLISH	SUPERVISOR
/Assets	Content	✓	✓	✓	✓	✓	
/Assets	Folder	✓	✓		✓	✓	
/aspTemp	Content	✓	✓	✓	✓	✓	
/aspTemp	Folder	✓	✓		✓	✓	
/Documentation	Content	✓	✓	✓	✓	✓	
/Documentation	Folder	✓	✓		✓	✓	
/Home	Content	✓	✓	✓	✓	✓	
/Home	Folder	✓	✓		✓	✓	
/Settings	Content	✓	✓	✓	✓	✓	
/Settings	Folder	✓	✓		✓	✓	
/Sites	Content	✓	✓	✓	✓	✓	
/Sites	Folder	✓	✓		✓	✓	
/System	Content	✓	✓	✓	✓	✓	
/System	Folder	✓	✓		✓	✓	
/Themes	Content	✓	✓	✓	✓	✓	
/Themes	Folder	✓	✓		✓	✓	

Figure 8.10. Check effective rules

8. Click **[Save]** in order to save the changes to the newly created group.
- If you want to discard changes, click **[Revert]**.

# 8.5 Adding, Deleting and Editing Rules

Rules define the rights on CoreMedia resources, that is content and folders.

## Adding Rules

- 1. Open the *Rules* tab of a group in the User Manager.

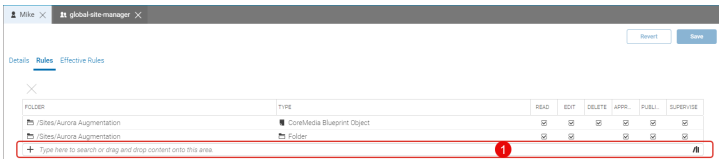


Figure 8.11. Adding rules

- 2. Rights are always defined below a specific folder. In order to select a folder for which you want to define a rule start typing in field (1) or click the library icon and drag a folder onto the field. The new rule is added to the list.

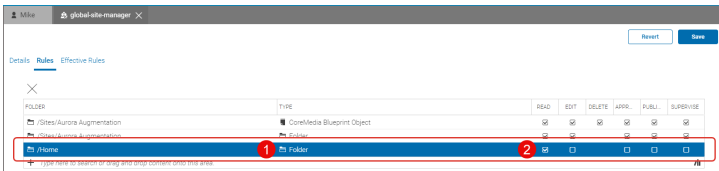


Figure 8.12. New rule in list

- 3. You can now edit the new rule. Click on the *TYPE* field to select the resource type (1) to which you want to apply the rule.

Now, you can attach rights. Check the checkbox (2) for the rights you want to grant.

- 4. If you want to check the effective rules for this group, open the *Effective Rules* tab.
- 5. Click **[Save]** to save your changes.

### Deleting Rules

1. Open the *Rules* tab of a group in the User Manager.
2. Select the rule you want to delete.
3. Click the "X" icon above the link list. The rule is deleted without any check.
4. If you want to restore the deleted rule, click **[Revert]** , otherwise click **[Save]** in order to save your changes.

Now, you have deleted the rule.

# 8.6 Defining Rights on Folders

As a user with supervise or administrator rights you can directly add rules to a folder in order to give other users editing rights.

- 1. From the context menu of a folder, select **Properties**. The **Properties** window opens up.

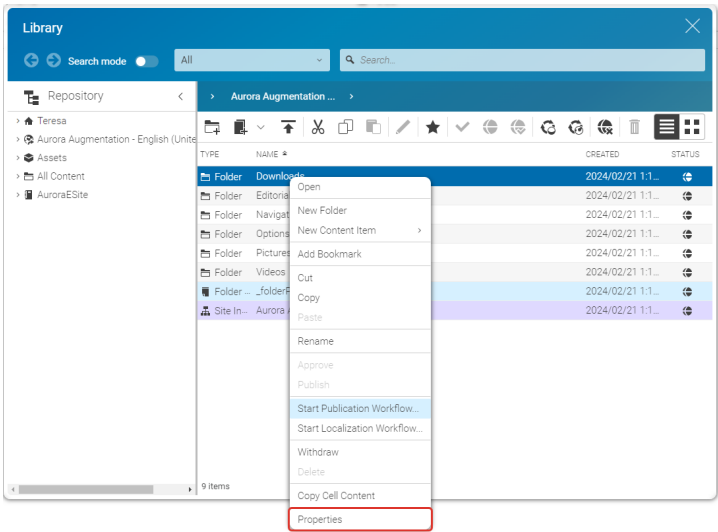


Figure 8.13. Open the Properties window

- 2. Select the *Permissions* tab. If there are already rules attached to the folder, you will see them and can edit them. When you want to revoke your own "Supervise" right, you will get a warning message which you have to confirm.
- 3. In the *Permissions* tab, click the + icon, in order to create a rule. You have to select a group to which the rule is added and select rights for these group on the selected folder.

In order to enable the editors to work on content, you need to add another rule for content inside the folder.



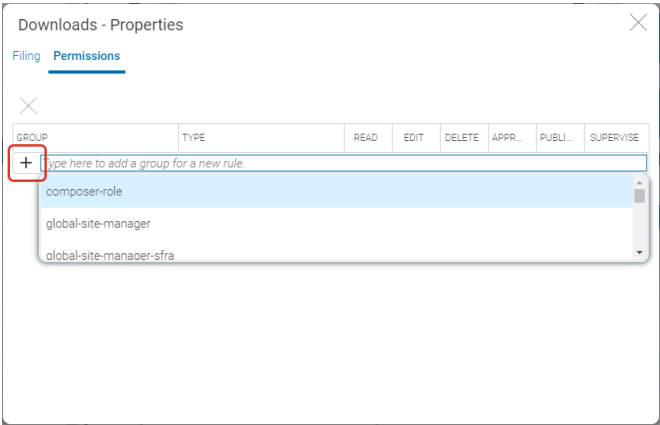


Figure 8.14. Add rules to folder

4. If you want to save the changes, click **[Save]** . If you want to revert your changes, click **[Reset]** .

**NOTE**

It is possible, that other users edit the permissions while you are working on it. When you try to save your changes and the other users have already saved their changes you will get a warning dialog and have to discard your changes.



## 9. CoreMedia Content as a Service (CaaS)

CoreMedia *Headless Server* is a CoreMedia component 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.

CoreMedia *Headless Server* provides an additional way of content delivery:

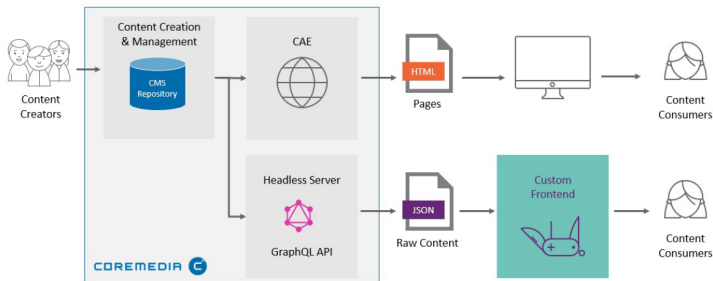


Figure 9.1. Headless Server overview

The *Headless Server* comes with the following feature set:

- Access through a [GraphQL](#) endpoint
- GraphQL **schema support** for CoreMedia content types with type inheritance (see [Chapter 4, Developing a Content Type Model](#) in *Content Server Manual* for details of CoreMedia content types).
- Support for **Spring EL** in GraphQL schemas
- Access to **CoreMedia business logic**
- Multi-Site/Language delivery
- Validity/Visibility of Content

## CoreMedia Content as a Service (CaaS) |

- Navigation and Page Grid support
- Responsive Images
- Rich Text Transformation
- Image Maps, Shoppable Videos, Teaser with multiple targets, Videos in Banners
- Full Text Search
- Dynamic Query Lists
- eCommerce integration via *CoreMedia Commerce Hub*
- Studio JSON Preview Client which integrates in CoreMedia Studio
- Deployment as a Spring Boot application

## 9.1 Endpoints of the *Headless Server*

For the *Headless Server* several endpoints are available.

### GraphQL

GraphQL is the standard endpoint of the *Headless Server* and available at `/graphql`.

It serves GraphQL requests as specified on [graphql.org](https://graphql.org).

### GraphiQL

GraphiQL is a graphical interactive in-browser GraphQL IDE. See the [GraphiQL GitHub repository](#) for details.

The GraphiQL endpoint is, by default, enabled for the *Headless Server* in preview mode and available at `/graphiql`.

### Swagger UI

Swagger UI is a tool to visualize and interact with REST resources. More information can be found at <https://swagger.io/tools/swagger-ui/>.

For the *Headless Server*, media objects are delivered via REST and can be inspected with Swagger UI.

Swagger UI is only available, if configured. Default, it is enabled for the *Headless Server* in preview mode and available at `/swagger-ui/index.html`.

### JSON Preview and Preview URL Service

The JSON Preview and corresponding Preview URL Service are only available in *Headless Server* preview mode and provide a preview integration into CoreMedia Studio. See [Section 9.2, “Preview” \[224\]](#) for details.

Endpoints for JSON Preview and Preview URL Service are `/preview` and `/previewurl`.

### REST

Persisted queries (see [Section 9.7, “Persisted Queries” \[256\]](#)) may be accessed by simple HTTP GET requests. As the persisted queries are customizable and freely definable by name, the endpoints are exposed dynamically relatively to the endpoint `/caas/v1/`.

All endpoints to persisted queries are documented automatically within the Swagger UI.

### Site Filter

A site filter restricts the access of GraphQL queries to content objects of one site only.

### Media Endpoint

The media endpoint serves all managed media files (BLOBs). It is available at `/caas/v1/media`.

## 9.2 Preview

Data delivered by CoreMedia *Headless Server* can be previewed in CoreMedia *Studio* by integrating a corresponding preview client.

A basic preview client that renders *Headless Server* data as a raw JSON data tree is available as part of the Blueprint workspace.

To display multiple Previews in Studio, the Multiple Previews Feature needs to be configured.

### 9.2.1 JSON Preview Client

The JSON Preview Client is available in the Maven module **json-preview-client** of the Blueprint workspace.

### 9.2.2 Custom Preview Client

For a custom preview client, a corresponding preview URL service needs to be set up. It should respond to preview URL requests for a given content ID with a URL where to fetch the actual preview HTML from the custom preview client.

## 9.3 Security

Depending on the frontend approach, the *Headless Server* may be fully or partially exposed to public access. Therefore, the *Headless Server* needs an effective protection.

GraphQL offers a self-descriptive approach to deliver data to client applications. This makes it easy to any client to use and visualize this data in any way, without the need to have an exact knowledge about the underlying data model, thus reducing the need for support. On the other hand, clients may request as much data as they wish, creating potentially high load on the server.

### WARNING

Because of the database character of any GraphQL endpoint, the publicly accessible content items should never contain any confidential data, like access credentials or user data.



Protecting the *Headless Server* can be realized by two general approaches:

- Externally, before the *Headless Server* is actually invoked, by using hardware (load balancers, firewalls), a web server (gateway) or a so-called backend-for-frontend approach.
- On the application layer of the *Headless Server* by means of configuration.

The external approach is usually very efficient. You may enforce certain access restrictions by employing some kind of authorization and/or authentication or define IP access restrictions. However, this approach implies, that the clients are in some kind 'known' by the server. If you want to allow accessing data by any client, this approach is hard to enforce.

Whenever it is not possible or not desirable to restrict access to known clients, you might use the application layer approach.

The *Headless Server* offers these options to employ security measures:

- Allowing only listed persisted GraphQL queries described in [Section 9.3.1, "Query Allow List for GraphQL Queries" \[226\]](#).
- Blocking of content items, especially `CMSSettings` content items, from delivery, using their repository path. See the deployment manual [Section 3.3, "Headless Server Properties"](#) in *Deployment Manual* for details about the configuration property `caas.graphql.repository-path-exclude-patterns`.

- Limiting the size of a search result described in [Section 9.3.2, “Limiting the Size of a Search Result” \[226\]](#).
- Limiting the depth of a GraphQL query described in [Section 9.3.3, “Limiting the Depth of a GraphQL Query” \[227\]](#).
- Limiting the complexity of a GraphQL query described in [Section 9.3.4, “Limiting the Complexity of a GraphQL Query” \[227\]](#).
- Enforce an execution timeout for GraphQL queries described in [Section 9.3.5, “Enforcing an Execution Timeout for GraphQL Queries” \[228\]](#).

All the above measures may be used to protect the server from expensive queries or malicious attacks.

### NOTE

In order to provide a certain amount of protection by default, the size of a search result is limited to 200 hits and the maximum query depth is set to 30. Especially on protected preview servers, these limits are possibly not desirable, while developing or testing GraphQL queries and therefore should be reconfigured to suite your needs.



## 9.3.1 Query Allow List for GraphQL Queries

A query allow list means that only the persisted queries that reside on the server are allowed to execute. All other GraphQL queries are denied.

The allow list may be enabled by setting the configuration property `caas.persisted-queries.allow-list` to `true`. See [Section 9.7.2, “Query Allow Listing” \[259\]](#) for more details.

## 9.3.2 Limiting the Size of a Search Result

Allowing unlimited result sizes on search queries is probably the easiest way to produce high load on the server. Therefore, limiting the size of a search result to a maximum value is almost imperative. Whenever the requested limit exceeds the maximum allowed limit, the requested limit is overwritten by the maximum value before the search query is invoked.



The maximum search result limit is enabled by setting the configuration property `caas.search.max-search-limit` to a value greater than `0`. The default maximum search limit is `200`.

When not requesting an explicit limit within a query, the default limit is `10`. If the configured maximum search limit is smaller than the default limit, it overwrites the default limit.

### 9.3.3 Limiting the Depth of a GraphQL Query

Any opening curly bracket in a GraphQL query marks the start of a new nesting level of the query. The depth of a query is then simply the deepest nested level. By limiting the depth of a query to a certain value, the size of the data is limited correspondingly. Furthermore, indefinite querying of circularly linked content is prevented. As the depth is calculated before actually invoking the query, the counter measure is quite efficient.

The depth limit is enabled by setting the configuration property `caas.graphql.max-query-depth` to a value greater than `0`. The default depth limit is `30`.

### 9.3.4 Limiting the Complexity of a GraphQL Query

The higher the complexity of a query is, the higher is the resulting potential load on the server. The complexity of a query may be limited by a `MaxQueryComplexityInstrumentation` which is provided by the `graphql-java` framework. By default, the complexity of a query is calculated by summing up the number of requested fields and nested levels. A more sophisticated complexity calculator may be added to the Spring configuration by implementing the `FieldComplexityCalculator` interface from `graphql-java`. Like the query depth, the complexity of a query is calculated before actually invoking the query.

The complexity limit can be enabled by setting the configuration property `caas.graphql.max-query-complexity` to a value greater than `0`. The default is `0` which means that this check is disabled.

## 9.3.5 Enforcing an Execution Timeout for GraphQL Queries

As a last resort, it is possible to enforce a maximum time to process a GraphQL query. Whenever that time is exceeded, a timeout kicks in, aborting the query execution. As at this point of time the query was already invoked, this type of counter measure should be considered as a last resort. If the server is under such a high load, instead of enforcing an execution timeout, please consider counter measures outside of the *Headless Server*, as mentioned above. Besides, if there is no malicious attack, the server resources like the number of processors, or RAM size may be sized too small. In such cases, raising limited resources or deploying another instance of the *Headless Server* may be fitting solutions.

The timeout is implemented by the `ExecutionTimeoutInstrumentation` provided by CoreMedia and bundled with the *Headless Server*. It can be enabled by setting the configuration property `caas.graphql.max-query-execution-time` to a value greater than `0`. The default value is `0` which means that no timeout is checked.

The timeout is set in milliseconds. A reasonable value may be `2000` or `3000` (that is, 2 or 3 seconds). Also keep in mind, that the first invocation of a query on a new instance of the *Headless Server* may take much longer than the follow-up queries due to caching effects.

## 9.3.6 MediaType Content Negotiation

The `MediaController` is responsible for the delivery of binary contents like images and other content types. For security reasons, the Spring framework sets the HTTP Content-Disposition response header to the static value `inline; filename=f.txt` for potentially insecure content types, for example, PDF files, unless it was specifically set previously.

This behaviour may produce undesirable results when downloading files via the `MediaController`, as the filename is anonymous and the content type is forced to the suffix `txt`, no matter what the real content type might be.

It is however possible to configure Spring to suppress this default behaviour for specific content types, using `CaasConfig`.

```
/**
 * Code example to suppress the default Content-Disposition header for
 * potentially insecure content types. Add to CaasConfig if necessary.
 */
@Override
```

```
public void configureContentNegotiation(
    ContentNegotiationConfigurer configurer
) {
    configurer.mediaType("pdf", MediaType.APPLICATION_PDF);
    configurer.mediaType("eps", new MediaType("application", "postscript"));
}
```

### *Example 9.1. Configuring Content Type Resolution for PDF and EPS Files*

Please see the original Spring Web MVC Documentation about Content Types for a more detailed insight about the security aspects and about so called reflected file download attacks (RFD).

Also refer to [Section 9.9, “Media Endpoint” \[263\]](#) about how the `MediaController` sets the Content-Disposition response header.

## 9.4 Search

The `headless-search` for the *Headless Server* encapsulates search related functionality like faceted and generic search, suggestions, dynamic query lists and their corresponding types.

It is part of the Blueprint Base module and contains a GraphQL schema extension within the file `search-schema.graphql`, Java code and Spring configuration.

If necessary, the `headless-search` can be deactivated by configuration properties. Note that it is possible, to deactivate the search schema extension explicitly, without deactivating the related code. This provides the possibility to add a customized version of the search schema. See the [Section 3.3.1, "Headless Server Spring Boot Properties"](#) in *Deployment Manual* for details.

To use Headless Server search, an existing Solr with an index created by a CAE Feeder needs to be provided.

### 9.4.1 Generic Search

Search related features are handled by the adapters `searchAdapter`, `facetedSearchAdapter` and `suggestionsAdapter`. The following functionality is supported:

- Full text search
- Paging
- Limit
- Filter by content type, optionally including their sub types
- Predefined sort fields with order
- Limitation to a site
- Valid from and valid to conditions are applied to search filters automatically
- Faceted search results
- Search suggestions

The following GraphQL query is a simple example for fetching a search result.

```
{
  content {
    search(query:"Perfect") {
      numFound
      result {
        name
      }
    }
  }
}
```

```
}  
}
```

Several parameters can be passed to the `SearchAdapter` to customize the search:

- `query`: The search query.
- `offset`: The offset.
- `limit`: The limit of search result.
- `docTypes`: Content types to restrict the search result.

Misspelled content types and invalid content types will cause a graphql error in the response. When passing an abstract content type, the subtypes are retrieved, if the parameter `includeSubTypes = true`. Passing an abstract content type with `includeSubTypes = false` will also cause a graphql error. The search result does not contain abstract content types, only the concrete sub types.

- `sortFields`: List of sort field with order, separated by '\_', in upper case, for example, `ID_ASC`.

The set of available sort fields in the schema is limited to the enum `SortFieldWithOrder` defined in the content schema: `ID`, `DOCUMENTTYPE`, `TITLE`, `TEASER_TITLE`, `MODIFICATION_DATE`, `CREATION_DATE`, `EXTERNALLY_DISPLAYED_DATE`. This enum can be extended in the schema by adding an available field with a sort order.

The available fields are defined in `SearchConstants#FIELDS`: `ID`, `DOCUMENTTYPE`, `NAVIGATION_PATHS`, `NOT_SEARCHABLE`, `SUBJECT_TAXONOMY`, `LOCATION_TAXONOMY`, `TITLE`, `TEASER_TITLE`, `TEASER_TEXT`, `KEYWORDS`, `MODIFICATION_DATE`, `CREATION_DATE`, `TEXTBODY`, `SEGMENT`, `COMMERCE_ITEMS`, `CONTEXTS`, `AUTHORS`, `HTML_DESCRIPTION`, `VALID_FROM`, `VALID_TO`, `EXTERNALLY_DISPLAYED_DATE`

To configure custom fields, a specific bean can be configured, see section below.

Possible order field values: `ASC`, `DESC`

- `siteId`: The `siteId` can be passed as parameter to restrict search per site.
- `includeSubTypes`: A Boolean flag, indicating to include the sub types of the given doc types in the search. Defaults to 'false'.

The query parameter supports the following syntax:

- The `+` and `-` characters are treated as "mandatory" and "prohibited" modifiers for terms.
- Quoted expressions, like "Foo Bar" are treated as a phrase
- An odd number of quote characters is evaluated as if there were no quote characters at all.

- The wildcard character '\*' supports the search for partial terms like 'frag\*', which would find, for example, the terms 'fragment' and 'fragile' as well. When used exclusively as a search query, the search is executed with all other search parameters but without an explicit search expression.

By default, the `SearchAdapter` employs `DefaultSearchServiceProvider`, which in turn uses the `caeSolrQueryBuilder` Spring bean. `caeSolrQueryBuilder` invokes searches on Solr on the `cmdismax` end-point. For details, see [Section 3.8.1, "Details of Language Processing Steps"](#) in *Search Manual*.

The used `SearchServiceProvider` is at the same time an `ExtensionPoint`, which can be implemented and provided by a plugin.

The following GraphQL query is a more complex example for fetching a search result.

```
{
  content {
    search(query: "Perfect", offset: 3, limit: 5, docTypes: ["CMArticle",
"CMPicture"], sortFields: [CREATION_DATE_ASC, MODIFICATION_DATE_ASC], siteId:
"abffe57734feeee", includeSubTypes: true) {
      numFound
      result {
        name
        type
      }
    }
  }
}
```

If `docTypes` or `limit` is not passed as parameter, the following search configuration is taken into account, which is read from CMS content using settings. See general search configuration for details in [Section 5.4.21, "Website Search"](#) in *Blueprint Developer Manual*.

- `searchDoctypeSelect`, `search.doctypeselect`: content types to restrict the search result
- `searchResultHitsPerPage`, `search.result.hitsPerPage`: limit of the search result

Valid from and valid to conditions are applied to search filters automatically.

## Faceted Search Results

The *Headless Server Search* is able to do a faceted search on configured facets on the Solr search index. *Headless Server* comes with preconfigured facets, for example, on the content type. See [Section 5.4.21, "Website Search"](#) in *Blueprint Developer Manual* on how to configure facets on the search index.

In contrast to the regular search without facets, the `facetedSearch` query requires the parameter `siteId` mandatorily. To issue a faceted search request,

the search query has to define the desired facets using the parameter `facetFilters`:

```
{
  content {
    facetedSearch(
      query: "*"
      siteId: "abffe57734feeee"
      facetLimit: 10
      facetFilters: [
        { facetAlias: "type", args: ["CMArticle"], excludeInFacet: false }
        { facetAlias: "subject", args: ["1234", "5678"] }
      ]
    ) {
      numFound
      facets {
        alias
        field
        values {
          query
          value
          hitCount
          facetContent {
            id
          }
        }
      }
    }
    result {
      id
      type
      ... on CMArticle {
        detailText {
          text
        }
      }
    }
  }
}
```

The facets can be found in the `facets` property of the search result. They provide information about the requested facets, the corresponding facet values and their count of content items where the facet occurred. If the facet value can be resolved to a content item, the content is provided as `Content_` in the property `facetContent`.

Use these parameters to issue a faceted search

- `query`: The search query.
- `offset`: The offset.
- `limit`: The limit of search result.
- `facetLimit`: Limits the size of facet values per facet. Defaults to studio config if set or 5 if not.
- `sortFields`: List of sort field with order, separated by '\_', in upper case, for example, `ID_ASC`.

The set of available sort fields in the schema is limited to the enum `SortFieldWithOrder` defined in the content schema: `ID`, `DOCUMENTTYPE`, `TITLE`, `TEASER_TITLE`, `MODIFICATION_DATE`, `CREATION_DATE`, `EXTERNALLY_DIS-`

PLAYED\_DATE. This enum can be extended in the schema by adding an available field with a sort order.

The available fields are defined in `SearchConstants#FIELDS`: ID, DOCUMENTTYPE, NAVIGATION\_PATHS, NOT\_SEARCHABLE, SUBJECT\_TAXONOMY, LOCATION\_TAXONOMY, TITLE, TEASER\_TITLE, TEASER\_TEXT, KEYWORDS, MODIFICATION\_DATE, CREATION\_DATE, TEXTBODY, SEGMENT, COMMERCE\_ITEMS, CONTEXTS, AUTHORS, HTML\_DESCRIPTION, VALID\_FROM, VALID\_TO, EXTERNALLY\_DISPLAYED\_DATE

To configure custom fields, a specific bean can be configured, see section below.

Possible order field values: ASC, DESC

- `siteld`: The siteld. The siteld is mandatory in order to retrieve the configured facets per site.
- `facetFilters`: List of `FacetFilter` input objects with one or more configured facets. Optionally with facet values to be excluded from the faceted search result. If no `FacetFilter` is given, all configured facets are calculated automatically.

The input type `FacetFilter` consists of these parameters:

- `facetAlias`: Mandatory name of a facet as configured.
- `filterValues`: Optional list of filter values for the given facet. The filter values are effectively a filter query on the configured field of the facet, e.g. `type` on the standard field `documenttype`.
- `excludeInFacet`: Defaults to `true`. If set false, the query clause with filter values is not excluded from facet calculation, resulting in a facet result with the given filter values only.
- `customFilterQueries`: Like the generic search, the facet search can also be extended by custom filter queries. See [Section 9.4.3, "Custom Filter Queries" \[238\]](#) for details.

### NOTE

A faceted search query is a non trivial and complex query. Be aware, that additional queries using the custom filter queries, might affect the search result and the facet calculation in unexpected manners.



By default, the `FacetedSearchAdapter` employs `DefaultFacetedSearchServiceProvider`, which in turn uses the `caeSolrQueryBuilder` Spring bean. `caeSolrQueryBuilder` invokes searches on Solr on the `cm` `dismax` endpoint. For details, see [Section 3.8.1, "Details of Language Processing Steps" in Search Manual](#).



The used `FacetedSearchServiceProvider` is at the same time an `ExtensionPoint`, which can be implemented and provided by a plugin.

# Search Suggestions

Suggestions are a very popular feature for any search on a website. Suggestions are calculated simultaneously and then provided as an optional list to choose from, thus relieving the user from typing the full search expression.

The *Headless Server* is able to provide suggestions for search query expressions.

```
{
  content {
    suggest(
      query: "sal"
    ) {
      value
      count
    }
  }
}
```

Use these parameters to issue a search suggestion query.

- `query`: The search query.
- `docTypes`: Content types to restrict the search result.

Misspelled content types and invalid content types will cause a graphql error in the response. When passing an abstract content type, the subtypes are retrieved, if the parameter `includeSubTypes = true`. Passing an abstract content type with `includeSubTypes = false` will also cause a graphql error. The search result does not contain abstract content types, only the concrete sub types.

- `siteId`: The `siteId` can be passed as parameter to restrict search per site.
- `includeSubTypes`: A Boolean flag, indicating to include the sub types of the given doc types in the search. Defaults to 'false'.
- `customFilterQueries`: Like the generic search, the search suggestions can also be extended by custom filter queries. See [Section 9.4.3, "Custom Filter Queries" \[238\]](#) for details.

By default, the `SuggestionAdapter` employs `DefaultSuggestionSearchServiceProvider`, which in turn uses the `suggestionsSolrQueryBuilder` Spring bean. `suggestionsSolrQueryBuilder` invokes searches on Solr on the `suggest` endpoint. For details, see [Section 3.8.1, "Details of Language Processing Steps" in Search Manual](#).

The used `SuggestionSearchServiceProvider` is at the same time an `ExtensionPoint`, which can be implemented and provided by a plugin.

## Configuration of custom SOLR fields

To configure a custom field of the SOLR index, a bean with qualifier `customSolrFields` can be added to the Spring context.

This bean of type `Map<String, String>` contains the custom field's name as a constant accessor and the field name in the SOLR index, e.g. `TITLE, title`.

This custom field can then be used, e.g. to apply a sort order.

The `customSolrFields` are applied to the `SolrQueryBuilder`.

The default SOLR fields are defined in the class `SearchConstants`, these are the default fields of the SOLR CAE index.

## Generic configuration

The connection to Solr is defined with `solr.url`

The search index is specified with property `caas.search.solr.collection`

Caching is only performed in live mode and can be configured with `caas.search.cache.seconds`

## Configuration of a custom index

If search should be performed on a custom SOLR index, the `SolrQueryBuilder` must be extended and configured. The following constructor arguments can be passed:

- `searchHandler`: the search handler, e.g. `/cmdismax`
- `filterQueryDefinitionMap`: a map containing filter query definitions to be used by custom filter queries
- `customFields`: custom fields of the SOLR index as map containing the field name and the SOLR field name, e.g. `TITLE, title`

## 9.4.2 Dynamic Query Lists

To use Dynamic Query Lists with *Headless Server*, *Headless Server* Search needs to be set up (see [Section 6.1.1, "Content Query Form"](#) in *Blueprint Developer Manual* for details about Dynamic Query List content).

Dynamic Query Lists are handled with the `queryListAdapter`. The following functionality is supported:

- Paging
- Limiting the result size
- Filter by predefined fields
- Sort by predefined fields

The following GraphQL query is a simple example for fetching data from a CM-QueryList content.

```
{
  content {
    queryList(id: "7692") {
      title
      items {
        ... on CMLinkable {
          title
        }
      }
    }
  }
}
```

The following parameter can be passed to the `QueryListAdapter` to customize the Dynamic Query List result:

- `offset`: The offset for paging. Available as `pagedItems` in graphql schema.

The following GraphQL query is a simple example for fetching paged data from a CMQueryList content.

```
{
  content {
    queryList(id: "7692") {
      title
      pagedItems(offset: 3) {
        title
      }
    }
  }
}
```

Dynamic Query List configuration is read from the content using configuration that can be applied in Studio.

General configuration:

<b>Content Types</b>	A selection of content types.
<b>Limit</b>	Limit of the Dynamic Query List items.
<b>Sort Field</b>	The field to sort on.
<b>Order</b>	The sort order

Search filter configuration:

<b>Authors</b>	The authors of the document.
<b>Context Documents</b>	The context of the document.
<b>Modification Date</b>	The modification date defines as interval.
<b>Location Tag</b>	The content is tagged with the given location tag.
<b>Subject Tag</b>	The content is tagged with the given subject tag.
<b>Tag Context</b>	The content is tagged with one of the tags of the query list's context.

Valid from and valid to conditions are applied to search filters automatically.

## Dynamic Query List Configuration

Caching for dynamic query lists is only performed in live mode and can be configured with `caas.search.cache.querylist-search-cache-for-seconds`

### 9.4.3 Custom Filter Queries

Generic search and dynamic query lists can be extended with custom filter queries, that are applied to the `fq` parameter of the Solr query.

Custom filter queries must be predefined in as an implementation of `CustomFilterQuery` and provided as a Spring bean, before they can be used. As `CustomFilterQuery` is an extension point also, custom filter queries may be provided as part of a plugin or directly, e.g. by `CaasConfig`.

## Definition of custom filter queries

The definition of a custom filter query consists of a query identifier and a function, that maps the field values to a Solr query.

- **Query Identifier:** a String value to identify the query. The graphql input type `FilterQueryArg` needs to be extended with the query identifier.
- **Mapping Function:** a function which takes a `List<String>` as argument and returns a String, that contains the Solr query in Solr syntax.

For example, a filter query definition could be defined with a query identifier `EXCLUDE_IDS` and a (here simplified) mapping function:

```
// Bean factory in a configuration class
@Bean
public CustomFilterQuery excludeIdsQuery() {
    return new CustomFilterQuery() {

        /**
         * The query identifier.
         */
        @Override
        public String getName() {
            return "EXCLUDE_IDS";
        }

        /**
         * The mapping function.
         */
        @Override
        public String apply(List<String> values) {
            return SearchQueryHelper
                .negatedQuery(
                    SearchQueryHelper
                        .orQuery(SearchConstants.FIELDS.ID.toString(), values)
                );
        }
    };
}
```

*Example 9.2. Example implementation of a custom filter query.*

In order to demonstrate the usage and possibilities, *Headless Server* comes with some out-of-the-box custom filter queries, namely:

- **TITLE\_OR:** Query for one or more exact search expressions on the title field of the index.
- **EXCLUDE\_IDS:** Exclude one or more content ids from the search result.
- **FRESHNESS:** Query for contents newer than the given date on the modification date field of the index.
- **LOC\_TAXONOMY\_OR:** Query for location taxonomy values.
- **SUBJ\_TAXONOMY\_OR:** Query for subject taxonomy values.

There are some help utilities in `SearchQueryHelper` to generate the Solr query in Solr syntax. Alternatively, the Solr query can also be given in direct Solr syntax.

### Apply custom filter queries

A custom filter query can be applied statically for all queries or dynamically for each graphql query.

#### Static custom filter queries

Static filter queries, that shall be applied to all Solr queries, can be passed to the corresponding `*AdapterFactories`, e.g. `SearchServiceAdapterFactory` or `QueryListAdapterFactory`. They are then added to all Solr queries automatically.

#### Dynamic custom filter queries

Dynamic filter queries, that are applied to a specific GraphQL query, can be added as query argument for generic search, faceted search, suggestions or dynamic query lists. The input format is defined via the built-in type `FilterQueryArg`

All custom filter queries are applied as `fq` (filter query) fragments to the Solr query.

This GraphQL query is an example for fetching a search result using the pre-defined custom filter queries `EXCLUDE_IDS` and `TITLE_OR`.

```
{
  {
    content {
      search(query: "", docTypes: ["CMArticle"], customFilterQueries:
[{"EXCLUDE_IDS: ["1234", "5678"]}, {"TITLE_OR: ["Make your dream come true",
"Eveningwear Trends"]}]}) {
        numFound
        result {
          id
          ... on CMArticle {
            title
          }
        }
      }
    }
  }
}
```

This GraphQL query is an example for fetching query list items using the pre-defined custom filter queries `EXCLUDE_IDS`.

```
{
  {
    content {
      queryList(id: "10") {
        ... on CMQueryList {
          id
          filteredItems(customFilterQueries: {EXCLUDE_IDS: ["1234", "5678"]})
        }
      }
    }
  }
}
```

```
... on CMLinkable {  
  id  
}  
}  
}  
}  
}  
}
```

## 9.5 Rich Text

Processing rich text content is a complex issue. The following two chapters describe, how the *Headless Server* handles different aspects of rich text processing.

### 9.5.1 Rich Text Output

Delivering CoreMedia RichText properties requires a transformation of the internally stored markup format into a format that can be serialized to JSON output and that matches the requirements of the client. This process is handled by a configurable set of *Rich Text Transformers* per RichTextTransformerRegistry. Each transformer handles a specific transformation aspect required by the client, for example:

- Generate a text only teaser from the first paragraph of a richtext property.
- Generate a full HTML representation of a detail text including embedded images and internal links.

Transformers are applied to the raw content of a GraphQL field on either of these types:

- `String`: A string representation of the complete Markup.
- `RichTextTree`: A custom scalar GraphQLType that defines a tree based representation of the markup.
- `[CMLocalized!]`: A list of all embedded content objects within the markup.

The output format may be specified by the transformation name in a GraphQL query with a view clause, where the name of the view is equivalent to the transformation name.

Please note, that the term 'view' is not connected in any way to the views of the CAE used for rendering the same content for different display purposes!

```
Syntax:
richtext-field-name {
  graphql-field-name(view: "transformation-name")
}

Example:
detailText{
  text: (view: "plainFirstParagraph")
}
```

Names of the currently predefined views are:



<b>default</b>	Delivers the complete content of the requested field, consisting of all embedded markup, links and images, for instance. This view is the default, if no view is specified.
<b>simplified</b>	Delivers the complete content of the requested field, where special embedded markup like links and images is replaced by a plain version.
<b>plainFirstParagraph</b>	Delivers the first paragraph of the requested field without any embedded markup.

Please also note that, for technical reasons, the delivered content in all views is always nested in a `<div>` tag

Rich text transformers are fully configurable via YAML configuration files. Each configuration defines the following elements:

<b>name</b>	The transformer's view name.
<b>elements</b>	List of rich text elements. Is included at the start of the YAML definition. Individual elements are accessed by reference from following handlers.
<b>classes</b>	List of known rich text CSS class names. Is included at the start of the YAML definition. Individual names are accessed by reference from following handlers.
<b>contexts</b>	List of processing contexts. Each context defines a list of handlers, which are responsible for: <ul style="list-style-type: none"><li>• Processing opening and closing elements.</li><li>• Processing text nodes.</li><li>• Transforming elements and attributes.</li></ul>
<b>initialContext</b>	Defines the root context.
<b>handlerSets</b>	An optional mapping of named handler lists. Allows grouping and reusing handlers in different contexts.

Writing a new transformer is easily accomplished. First, create a YAML text file and place it in the Blueprint in the folder `resources/richtext`. The name of the file should match the name of the view used later in your GraphQL queries, for example a transformer named 'myView':

```
resources/richtext/myView.yml
```

As a starting point, add this basic content to your transformer file:

```
#!/import file=includes/elements.yml
#!/import file=includes/classes.yml
#!/import file=includes/attributes.yml

name: myView
contexts:
  - &root !RootContext
    name: root
    handlers:
      - !Handler
        eventMatcher: !Matcher {qname: }
        outputHandler: !ElementWriter {writeCharacters: true}
    initialContext: *root
```

Note that the file name (without the suffix) matches the 'name' property. As mentioned above, any transformer consists of the top level YAML properties 'name', 'elements', 'classes', 'contexts', 'handlerSets' and 'initialContext', which are all included in this basic example file.

When writing a configuration in YAML style, the indentation is most important. For a reference about YAML you may refer to <https://yaml.org/>.

### 9.5.2 Using RichTextAdapters for Different Rich Text Grammars

The *Headless Server* comes with an architecture to parse different flavors of rich text, including an out of the box RichTextAdapter to parse and transform the well known CoreMedia rich text grammar. The architecture allows customizing both, the grammar to be parsed and the underlying parsing technology, using standard Spring Boot beans.

The content repository delivers rich text as objects of type Markup, whereas the content schema declares a custom scalar type RichTextTree on all fields of the type Markup. The underlying architecture of graphql-java requires registering an implementation of the Coercing interface for a declared custom scalar type (RichTextTree). This is done in the config class CaasConfig by adding the scalar type and its conversion type Map and creating a bean of type GraphQLScalarType, which takes the Coercing implementation. By doing this, graphql-java now always expects a Map<String, Object> object when resolving fields of the scalar type RichTextTree.

With this kind of registration, only one Coercing class per scalar is possible. To overcome this limitation, CoreMedia has added a mechanism to invoke custom classes to handle different grammar types and to use any type of parsing/transformation technology.

## 9.6 eCommerce Extension

All eCommerce functionality of the *Headless Server* is bundled within the Blueprint module `headless-server-ec-augmentation`. It contains GraphQL schema extension files, Java code and Spring configuration to implement this schema extension. The extension allows clients to issue GraphQL queries for augmentation data for categories, products, external commerce pages and product lists.

The GraphQL schema extension contains commerce specific types and support for product and category augmentations.

The schema extension uses the GraphQL extension mechanism to add a new field `commerce` of type `CommerceRoot` to the query root object. This API may use an underlying Commerce Hub connection to the commerce system. Some of the commerce related calls can also be found below `content` as long as they do not need an underlying Commerce Hub connection.

### No Commerce Data

The eCommerce extension does not provide access to pure eCommerce related data like catalogs, categories and products. Instead the *Headless Server* provides augmentation data for categories, products, external commerce pages, product lists and navigation. Pure eCommerce data should be retrieved from the eCommerce system itself. In order to use the *Headless Server* in ecommerce projects with GraphQL, projects should use a schema gateway to combine both schemas (CoreMedia Headless Server and commerce system) to one combined graph. It is also possible to let a client talk to both backends in parallel, depending on the degree of integration needed.



## 9.6.1 Headless Commerce Integration Architecture

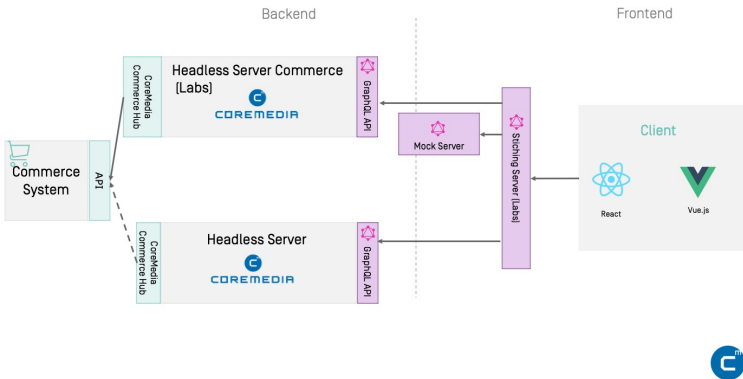


Figure 9.2. Headless Commerce Integration Example

The diagram shows an example architecture of a commerce integration with the CoreMedia Headless Server. In addition to the CoreMedia Headless Server, other CoreMedia labs components are used in the example setup. These labs components cannot be used in real projects without customization.

- *Client [Labs]*

Spark is a CoreMedia example application based on React, TypeScript and the Headless Server of CoreMedia Content Cloud. It uses the stitching server as single data endpoint for commerce and content data. The CoreMedia Spark example application is no official CoreMedia product, but is available as a CoreMedia labs project. See <https://github.com/CoreMedia/coremedia-headless-client-react>.

- *Stitching Server [Labs]*

The Stitching Server merges the GraphQL-Endpoints of the Headless Commerce Server and Headless Content Server and dispatches incoming GraphQL-Queries to the corresponding endpoints. The Stitching Server is no official CoreMedia product, but is part of the Spark Workspace and available as a CoreMedia labs project.

- *Mock Server [Labs]*

If there is no commerce system available for frontend development, the Mock Server can be used to provide commerce data to the client. The data can be recorded and replayed and it is stored in the file system. The Mock Server is no official CoreMedia product, but is part of the Spark Workspace and available as a CoreMedia labs project.

- *Headless Server Commerce [Labs]*

The Headless Commerce Server is an example GraphQL endpoint for the commerce data. The server establishes a UAPI connection to a Content Server and gRPC connections to configured CoreMedia Commerce Adapters (Commerce Hub). Headless Server Commerce is no official CoreMedia product, but is available as a CoreMedia labs project. A component that corresponds to the Headless Commerce Server component is obsolete if your commerce system offers a GraphQL endpoint on its own. See <https://github.com/Core-Media/coremedia-headless-commerce>

- *CoreMedia Headless Server*

The *Headless Server* serves as GraphQL endpoint for pure content data. It also provides access to content, which augments commerce products and categories. The server establishes a UAPI connection to a Content Server. Although it is not the endpoint for commerce data, the *Headless Server* still uses an underlying Commerce Hub connection to load hierarchical catalog information from a connected commerce system (see [Section 9.6.2, “Augmentation” \[247\]](#)).

- *Commerce System*

The commerce system provides access to commerce data. If the commerce system offers its own GraphQL API it should be used directly.

## 9.6.2 Augmentation

Categories, products and pages from the eCommerce system can be augmented with content from the CoreMedia CMS. This includes mapping media content such as pictures, videos and downloads to categories and products, as well as augmenting pages, categories and products with specific content objects (see [Section 6.2.3, “Adding CMS Content to Your Shop”](#) in *Studio User Manual*).

### 9.6.2.1 Categories and Products Mapped to Media Content

CMS media content can be associated with products and categories by adding the product or category to the `Associated Catalog Items` form field in

the `Metadata` tab within Studio (see [Section 6.2.3, “Adding CMS Content to Your Shop”](#) in *Studio User Manual*).

To query this media content, the GraphQL type `Augmentation` contains the fields `picture`, `pictures`, `video`, `videos`, `media` and `downloads`, where the singular forms just retrieve the first picture or video in the list.

For example, pictures associated with a product may be queried as follows:

```
{
  content {
    productAugmentationBySite(externalId: "PC ORANGE TEA", breadcrumb:
["PC_DELI", "PC_ToDrink"], siteId: "99c8ef576f385bc322564d5694df6fc2") {
      commerceRef {
        externalId
        siteId
        locale
      }
      pictures {
        name
        uriTemplate
        crops {
          name
          aspectRatio {
            width
            height
          }
          sizes {
            width
            height
          }
        }
      }
    }
  }
}
```

If any picture is associated with the given product in the CMS (by the aforementioned mapping in Studio), the returned URLs point to the corresponding picture.

The `picture` and `pictures` fields have the types `CMPicture` and `[CMPicture]!` types, respectively. This way, the full functionality of CMS pictures may be used to enrich the product presentation, such as picture variants with responsive image URI templates (see [Section 9.9, “Media Endpoint” \[263\]](#)).

As an alternative, the more general `visuals` field may be used to query for pictures, videos and other visual content as a single list.

Any pictures or thumbnails defined on the commerce side should be retrieved from the commerce system endpoint.

### 9.6.2.2 Augmented Categories and Products

Categories and products can be augmented with content of type `CMExternalChannel` and `CMExternalProduct`, respectively. These content objects are created in Studio, if you choose the menu item `Augment Category` for

`categories` or `Augment Product` for products. See [Section 6.2.3, “Adding CMS Content to Your Shop”](#) in *Studio User Manual* for more details.

If a product is augmented, an augmenting content is created and the product/category is linked internally via the `externalId` field. If you query the augmentation for the product/category from the *Headless Server*, you receive a `ProductAugmentation` or `CategoryAugmentation` respectively. An `Augmentation` type provides access to page grid placements, linked assets or the augmenting content itself. Note that not every product/category is augmented and therefore the `content` field can be null.

In contrast to plain content related page grid placements, page grids for augmentations are inherited along the commerce navigation hierarchy. For example, a product variant cannot be augmented itself, instead it inherits placements from the parent product, a product inherits placements from its category, which in turn inherits placements from its parent category or channel, all up the commerce navigation hierarchy.

There are two ways to do augmentation queries:

- `CommerceRoot`
- `ContentRoot`

### Query for augmenting content with CommerceRoot

To retrieve the hierarchy information of the category tree the *Headless Server* uses a connection to a commerce adapter under the hood. These augmentation queries can be found below the `CommerceRoot` (see section [Section 9.6.1, “Headless Commerce Integration Architecture”](#) [246]).

```
{
  commerce {
    productAugmentationBySite(externalId: "PC_BRITISH_TEA", siteId:
"99c8ef576f385bc322564d5694df6fc2") {
      pdpPagegrid {
        placements(names: ["header", "additional"]) {
          ...
        }
      }
    }
  }
}
```

### Query for augmenting content with ContentRoot

In contrast, the *Headless Server* also offers augmentation queries below the `ContentRoot`. These queries do not rely on an underlying commerce connection, but need to receive hierarchy parameter from the client. In case the com-

merce connection is sometimes slow, it can also slow down the augmentation queries of the *Headless Server*.

```
{
  content {
    productAugmentationBySite(externalId: "PC BRITISH TEA", breadcrumb:
["PC_DELI", "PC_ToDrink"], siteId: "99c8ef576f385bc322564d5694df6fc2") {
      pdpPagegrid {
        placements(names: ["header", "additional"]) {
          ...
        }
      }
    }
  }
}
```

You might have noticed the difference between the call below `content` and `commerce`. The call below `content` needs an additional `breadcrumb` parameter, as this query cannot use an underlying Commerce Hub connection to automatically resolve the category hierarchy of the requested product. The breadcrumb information is used to search for augmented categories in the content repository.

An `Augmentation` type provides access to page grid placements of categories, products and product variants. For categories, the placements of the ordinary page grid are retrieved, while for products the Product Detail Page (PDP) and the corresponding `pdpPagegrid` is used. Product variants simply inherit all placements from their parent product.

### NOTE

It is recommended to use the augmentation API below the `ContentRoot` because it is the future-proof solution with less calls and better decoupling.



The placements within a page grid can be retrieved in whole, including the complete grid structure with grid rows. Alternatively, a plain list of placements can be retrieved, optionally filtered by placement names. In the following example, only the placements `"header"` and `"additional"` are retrieved for a product:

```
{
  content {
    productAugmentationBySite(externalId: "PC BRITISH TEA", breadcrumb:
["PC_DELI", "PC_ToDrink"], siteId: "99c8ef576f385bc322564d5694df6fc2") {
      commerceRef {
        externalId
        siteId
        locale
      }
      content {
        repositoryPath
        ... on CMTeasable {
          title
          teaserText
        }
      }
    }
  }
}
```



```
}
  pdpPagegrid {
    placements(names: ["header", "additional"]) {
      name
      items {
        name
        type
        ... on CMTeasable {
          teaserTitle
          teaserText
          picture {
            uriTemplate
          }
        }
      }
    }
  }
}
```

In this example, you also query the `title` and `teaserText` fields of an associated `content`. Note that this `content` field is only non-null if this product is actually augmented. The same is true for the `content` in category augmentations – that field is only non-null if exactly this category is augmented, the field value is not inherited from the parent category.

### 9.6.2.3 Augmented Pages

Pages within the eCommerce system can be augmented with `CMExternalPage` content objects (see [Section 6.2.3.6, “Adding Content to Other Pages”](#) in *Studio User Manual*). The `commerce` root object offers a field `externalPage` which allows querying the CMS page content given a page ID and a site ID. The following example query retrieves the `header` and `main` placements from the `CMExternalPage` associated with the `about-us` page:

```
{
  commerce {
    externalPage(externalId: "about-us", siteId: "sfra-en-gb") {
      externalId
      name
      grid {
        placements(names: ["header", "main"]) {
          name
          items {
            name
            type
          }
        }
      }
    }
  }
}
```

```
}
}
```

## 9.6.3 Product Lists

Product lists are handled with the `productListAdapter` (see [Section 6.2.2.2, “Adding a Product List”](#) in *Studio User Manual*). The following functionality is supported:

- Paging
- Limiting the result size
- Filter by Subcategories with a specific value

The following GraphQL query is a simple example for fetching data from a CM-ProductList content.

```
{
  content {
    productList(id: "856") {
      items {
        ... on CMTeasable {
          teaserTitle
          teaserText
        }
        ... on ProductRef {
          externalId
        }
      }
    }
  }
}
```

Product List configuration is done in CoreMedia Studio, such as:

- First Displayed Position: The position of the first item to be displayed (for paging)
- Limit: Limit of the products in the Product List
- Order: The sort order

The results of the query are automatically filtered for Valid from and valid to conditions.

## Product List Cache Configuration

Caching for Product lists is only performed in live mode and the caching time can be configured with `caas.search.cache.querylist-search-cache-for-seconds`

## 9.6.4 References to Products and Categories

The *Headless Server* does not provide access to purely commerce data directly. Instead the schema includes the types `CategoryRef` and `ProductRef`, which represent a link to a category or a product respectively. Links from CMS contents to commerce objects can be accessed via a `productRef` for content of type `CMExternalProduct` or `categoryRef` for content of type `CMExternalChannel`.

```
{
  content {
    content(id: "3240") {
      ...on CMExternalProduct {
        repositoryPath
        productRef {
          externalId
          locale
          storeId
        }
      }
    }
  }
}
```

The `CategoryRef` can be used to be resolved externally into a category, the `ProductRef` can be resolved into a product. This can be done via schema stitching or directly within a headless client application.

For example a product list query would look like this:

```
{
  content {
    content(id: "850") {
      ... on CMProductList {
        items {
          ... on CMTeasable {
            teaserTitle
          }
          ... on CommerceRef {
            externalId
            storeId
            locale
          }
        }
      }
    }
  }
}
```

And here the data retrieved:

```
{
  "data": {
    "content": {
      "content": {
        "items": [
          {
```

```

        "externalId": "AuroraWMDRS-1",
        "storeId": "1",
        "locale": "en-US"
    },
    {
        "externalId": "AuroraWMDRS-4",
        "storeId": "1",
        "locale": "en-US"
    },
    {
        "teaserTitle": "Find your personal style"
    },
    {
        "teaserTitle": "Editorial Blog"
    },
    {
        "externalId": "AuroraWMDRS-23",
        "storeId": "1",
        "locale": "en-US"
    },
    {
        "externalId": "AuroraWMDRS-24",
        "storeId": "1",
        "locale": "en-US"
    }
    ]
}
}
}
}

```

A `CommerceRef` includes the data needed to load the product itself from the commerce system again.

## 9.6.5 eCommerce Setup and Configuration

Although the *Headless Server* does not deliver catalog data, it still needs an underlying commerce connection to resolve page grids inherited along the commerce category hierarchy, extend the commerce navigation and provide dynamic product lists managed in Studio. Therefore a running Commerce Hub is required. In addition, at least one properly configured Commerce Adapter is required in the *Headless Server* app.

Depending on your system setup, this may be any combination of

```

spring.grpc.client.channels.sfcc.address
spring.grpc.client.channels.hybris.address
spring.grpc.client.channels.commercetools.address
spring.grpc.client.channels.wcs.address

```

For catalog image URLs, a site mapping has to be configured in the same way as for the CAE, for instance

- For a local CAE:  
`blueprint.site.mapping.calista=http://localhost:49080`
- for Docker deployment:

```
BLUEPRINT_SITE_MAPPING_CALISTA: //preview.${ENVIRON  
MENT_FQDN:-docker.localhost}
```

## 9.7 Persisted Queries

Persisted Queries allow clients to issue GraphQL queries without transferring the whole (potentially long) query string at each request. Instead, clients pass a short ID or hash of the query string. The actual query string is stored on the server side, either by loading it at server startup, or by a client upload as part of an *Automatic Persisted Query*.

Persisted Queries have the following advantages:

- Reduced bandwidth

The payload of the request is generally reduced.

- Better CDN cacheability

Clients can use HTTP GET requests even for large queries.

- Reduced latency

Using HTTP GET makes it easy to avoid CORS preflight requests issued by a browser client (HTTP OPTIONS requests).

- Query allow list

Client queries may be restricted to the queries already known to the server, blocking potentially malicious queries.

Several GraphQL client frameworks support persisted queries, including Apollo Client and Relay. The CoreMedia *Headless Server* allows you to leverage this advanced GraphQL feature.

- [Section 9.7.1, “Loading Persisted Queries at Server Startup” \[257\]](#) describes how to set up the *Headless Server* to load persisted queries at startup time. This allows for the query allow list if the set of queries issued by clients is known in advance.
- [Section 9.7.2, “Query Allow Listing” \[259\]](#) describes the query allow list. That is, only queries loaded in the server during startup can be executed.
- [Section 9.7.3, “Apollo Automatic Persisted Queries” \[260\]](#) describes a more flexible approach called Automatic Persisted Queries. Automatic Persisted Queries allow clients to upload persisted queries to the server at runtime.

## 9.7.1 Loading Persisted Queries at Server Startup

Resource files containing GraphQL queries can be loaded into the Headless Server at server start up time, turning these queries into persisted queries.

Currently, three different resource file formats are supported for persisted queries, namely plain GraphQL files and JSON maps in Apollo and Relay format.

### 9.7.1.1 Defining Persisted Queries in Plain GraphQL

All resources matching the pattern configured with the property `caas.persisted-queries.query-resources-pattern` are loaded as persisted queries, one query per resource file. The filename without extension serves as the query ID. The pattern must be suitable for a Spring `PathMatchingResourcePatternResolver` which is used to load these resources.

The default pattern is `classpath:graphql/queries/*.graphql`, which means that all resource files within the `graphql/queries` directory are loaded if they have the `graphql` file extension.

Actually, not all resource files matching this pattern might be loaded - there is a configuration property `caas.persisted-queries.exclude-file-name-pattern` that specifies a regular expression for resource files to be ignored.

This pattern defaults to `.*Fragment(s)?.*graphql` which is useful to skip resource files holding reusable query fragments. These fragments may then be included into a query file by means of the `#import` directive. The following is an example query including fragments from the resource `referenceFragments.graphql`:

```
query ArticleQuery($id: String!) {
  content {
    article(id: $id) {
      ... Reference
      title
      detailText
      teaserTitle
      teaserText
    }
  }
}
```

```
}
#import "./referenceFragments.graphql"
```

If this query is saved in a resource file with the name `article.graphql`, the query will have the ID `article`. Therefore, you may now send an HTTP GET request with just this ID instead of the query string (mind URL encoding):

```
wget -q -O - 'http://myheadlessserver:41180/graphql \
?query=PersistedQueryMarker \
&extensions={"persistedQuery": {"version": 1,"queryId": "article"}} \
&variables={"id":"1556"}'
```

## 9.7.1.2 Defining Persisted Query Maps in Apollo Format

The [Apollo client](#) tool extracts GraphQL queries from your client code and generates a JSON file in the following form:

```
{
  "version": 2,
  "operations": [
    {
      "signature":
"88a2611edf717d47e91712e57f652aed0efb8ffa3190466aa05ce448468203c5",
      "document": "query ArticleQuery(...) {...}",
      ...
    }, {
      "signature":
"64cff55bclc8bfc2e6f8522aa4481bebee33eb7f1d9d9a3c8af12fc2e2aa2a9b",
      "document": "query PageQuery(...) {...}",
      ...
    }, ...
  ]
}
```

This JSON file can then be used by your client and the *Headless Server* for query allow list (see [Section 9.7.2, “Query Allow Listing” \[259\]](#)).

For the *Headless Server*, the JSON file must be accessible at server startup time as a resource resolvable by a [Spring PathMatchingResourcePatternResolver](#). One way to do this is to transfer the JSON file to the *Headless Server* workspace for inclusion at build time as a Java resource file.

By default, the *Headless Server* looks for Apollo query maps at locations specified by the configuration property `caas.persisted-queries.apollo-query-map-resources-pattern`, which defaults to `classpath:graphql/queries/apollo*.json`.



### 9.7.1.3 Defining Persisted Query Maps in Relay Format

The [Relay Compiler](#) may be asked to extract GraphQL queries from your client code and to generate a JSON file containing a map from query IDs (which are MD5 hashes) to query strings, for example:

```
{
  "33c07385fca167d81c2906b4f2ada3ac": "query AppArticleQuery(...) {...}}",
  "d614bb0396056705ef5a00815b828076": "query AppPageQuery(...) {...}}",
  ...
}
```

This map can then be used by your client and the *Headless Server* for query allow list (see next section).

For the *Headless Server*, the JSON map must be accessible at server startup time as a resource resolvable by a [Spring PathMatchingResourcePatternResolver](#). One way to do this is to transfer the JSON file to the *Headless Server* workspace for inclusion at build time as a Java resource file. By default, for the *Headless Server*, the JSON map must be transferred to the *Headless Server* workspace to be included at build time. The *Headless Server* looks for Apollo query maps at locations specified by the configuration property `caas.persisted-queries.relay-query-map-resources-pattern`, which defaults to `classpath:graphql/queries/relay*.json`.

## 9.7.2 Query Allow Listing

Registering queries in an allow list is a way to make the *Headless Server* more robust against potentially malicious (for example, expensive) queries. When allow-list is turned on, the *Headless Server* will execute only the queries loaded into the allow list of the server during startup. All other queries will be rejected with an error message in the JSON response.

The allow list in the *Headless Server* may be turned on by setting the configuration property `caas.persisted-queries.allow-list` to `true`.

Queries issued by clients do not need to match exactly those in the allow list. It suffices if their *normal form* is equal to the normal form of an allowed query. This is realized by means of the [QueryNormalizer](#) which transforms a GraphQL query string into a normal form, where definitions and fields follow a specific order (for example, lexicographically) and whitespace is minimized.

The allow list is recommended for projects which expose a GraphQL service for some dedicated clients for which the set of queries issued by the clients is

known in advance. Usually, you will want to turn allow-list off for your development environment so that front end developers can utilize the full flexibility of GraphQL. Once client development has finished, the queries can be extracted from the client code and transferred to the production environment where allow-list is turned on.

## 9.7.3 Apollo Automatic Persisted Queries

The allow list is a good and recommended option for services where the exact set of queries that clients may issue is known in advance (see [Section 9.7.2, “Query Allow Listing” \[259\]](#)). It is not an option for services which expose a generic API in GraphQL terms, such as the [Github API](#). For such a service, allowing only a predefined set of queries would be far too restrictive, so potentially malicious queries must be detected by other means than a simple allow list.

The Automatic Persisted Queries protocol proposed by Apollo has been designed for such services. It provides a way to take advantage of persisted queries (but without an allow list) without losing the flexibility of the original GraphQL service.

The main idea of Automatic Persisted Queries is an optimistic request passing the SHA256 hash of the query instead of the query string itself. If the query is already known to the server, the server executes the query as normal. If the query is not known to the server, it answers with a `PersistedQueryNotFound` error. The client then reissues the request, this time passing the query string along with the hash. The next time, if the same or another client issues an optimistic request with the same hash, the server can process the query and respond with the result right away.

Automatic Persisted Queries in the *Headless Server* are turned on by default. They may be turned off by setting the configuration property `caas.persisted-queries.automatic` to `false`. However, uploading arbitrary queries is disabled anyway if allow-list is turned on. Then, uploading queries is still supported for queries with a normal form equal to the normal form of those in the allow list.

## 9.8 Site Filter

Many relational database systems offer a "view" feature. A view provides an easy way to "see" only data, which is relevant for a certain use case. The *Headless Server* adopts this concept, to provide a filter to a specific site. Therefore, a site filter restricts the access of a GraphQL query to content objects of only one site.

In a scenario where CoreMedia is used to host a multitude of sites, like a site for each brand, prefiltered content might make it easier for frontend developers to develop a frontend client for one specific brand. Furthermore, potential copyright problems for media content like pictures, for example, or an unintentional mixup of contents belonging to different sites, are prevented effectively.

A site filter is invoked simply by putting the homepage segment in front of the standard GraphQL endpoint or any of the REST endpoints mapped to persisted GraphQL queries.

Given a site with a homepage segment of 'corporate-de-de', a site filter would result in these additional endpoints:

```
# generic access pattern to GraphQL with a site filter prefix
# http://[hostname]/[homepage-segment]/graphql
http://[hostname]/corporate-de-de/graphql

# generic access pattern to a REST endpoint with a site filter prefix
# http://[hostname]/[homepage-segment]/caas/v1/[restendpoint]
#
# given, there is a defined REST endpoint to /article,
# incl a correspondingly named persisted query
http://[hostname]/corporate-de-de/caas/v1/article/[id]
```

A complete listing of all existing site specific endpoints and its site ids can be acquired via the additional custom actuator endpoint at `/actuator/siteRestrictedEndpoints` or via the Swagger UI. The list via the Swagger UI only reflects the state at server start. As the list of site specific endpoints may change during runtime of the headless server, those changes are only available via the custom actuator endpoint.

The site filter access is enabled by default. If the site filter access is not desired, the feature can be disabled by adding its autoconfiguration class to SpringBoots exclude list, e.g. in an environment variable.

```
SPRING_AUTOCONFIGURE_EXCLUDE=com.coremedia.caas.web.view.impl.ViewAutoConfiguration
```

### Limitations

A site filter restricts the access to contents which belong to one site. This is accomplished without the use of users, groups or access rights. Using the standard endpoints (/graphql) without a site filter, it is still possible to access any data of any site! If you want to prevent the full access, please consider a corresponding access rule in your gateway web server.



## 9.9 Media Endpoint

The media endpoint provides access to all media files (blobs), managed by the CMS. The endpoint supports image transformation in terms of precalculated crop sizes and supported image formats (see [Section 9.5.3, “Image Cropping and Image Transformation”](#) in *Studio Developer Manual* for details about crops). The URL to a managed media file is usually retrieved by means of a GraphQL query.

The following examples show, how you retrieve the URL of images and media files via a GraphQL query.

```
{
  content {
    picture(id: "1904") {
      id
      name
      uriTemplate
      crops {
        name
        sizes {
          width
        }
      }
    }
  }
}
```

*Example 9.3. Retrieving the URI template of a picture*

```
{
  content {
    picture(id: "1904") {
      id
      name
      uriTemplate(imageFormat: PNG)
      crops {
        name
        sizes {
          width
        }
      }
    }
  }
}
```

*Example 9.4. Retrieving the URI template of a picture with an alternative image format*

```
{
  content {
    picture(id: "1904") {
      id
      name
      data {
        uri
      }
    }
  }
}
```

```

    }
    fullyQualifiedUrl
  }
}

```

*Example 9.5. Retrieving the URI or the fully qualified URL of the original file of a picture*

It is important to note that the URLs accepted by the media endpoint (blob delivery) should always match the urls retrieved via GraphQL (link building). The link building is performed by Link Composers and Adapters and then woven into the GraphQL schema using the `fetch` directive to invoke the named bean `uriLinkComposer`. For details about Link Composers and Adapters please refer to the corresponding sections [???](#) and [???](#).

## 9.9.1 Media Endpoint URLs

The media endpoint consists of the following distinct endpoints:

- Endpoint for images with crops and width.
- Endpoint for images with crops and width, format transformation and file name.
- Endpoint for generic media files.

### Endpoint for images with crops and width

The first endpoint requests an image by means of the name of the crop and the desired width. The structure of the URI template is as follows:

```
/caas/v1/media/{mediaId}/{propertyName}/{hash}/{cropName}/{width}
```

The supported crop names and widths can be retrieved as part of the query for the `uriTemplate` (see [Section 9.9, “Media Endpoint” \[263\]](#)). The placeholders 'cropName' and 'width' must be replaced by a valid combination of the supported values. Trying to request an invalid 'cropName' or 'width' will result in an `HTTP 404 Not Found` error.

### Endpoint for images with crops and width, format transformation and file name

The second endpoint to request images additionally supports on-the-fly image format transformation and the original file name. The structure of the URI template is as follows:

```
/caas/v1/media/{mediaId}/{propertyName}/{hash}/{cropName}/{width}/{filename}
```

The image format is specified by the file extension in the 'filename'. The format transformation is triggered by replacing the file extension with one of the supported image formats 'jpg', 'jpeg', 'png' or 'gif', or by directly requesting the respective `uriTemplate` like shown in the examples in [Section 9.9, “Media Endpoint” \[263\]](#). Requesting an unsupported format will result in an `HTTP 400 Bad Request` error.

### Endpoint for generic media files

The third endpoint is the most generic. It provides access to any media file which is managed by the CMS. The structure of the URI template is as follows:

```
/caas/v1/media/{mediaId}/{propertyPath}/{hash}[/{filename}]
```

Note that the `filename` is optional.

You can comfortably explore the described endpoints using the Swagger UI provided with the overview page.

### Content Disposition Header

Whenever a media file is requested with its correct filename (including the file extension), the HTTP header `Content-Disposition` will be set to `inline; filename=<the-original-file-name>`.

### Placeholders of the media endpoints

<b>mediaId</b>	The content ID of the media/picture.
<b>propertyName</b>	The name of the property, where the blob is stored, usually <code>data</code> .
<b>propertyPath</b>	The name of the property where the blob is stored or the full property path, in case the blob ist stored in a struct.
<b>hash</b>	The hash of the blob. Usually queried via GraphQL.
<b>cropName</b>	The name of an existing crop of an image. Usually queried via GraphQL.
<b>width</b>	An existing width belonging to the crop name. Usually queried via GraphQL.

**filename**

The file name of a media file, including its file extension. Usually queried via GraphQL.



## 9.10 REST Access to GraphQL

Although CoreMedia recommends using the GraphQL endpoint to develop modern client applications, it may be desirable to run a client application using a REST API, for different reasons:

- A REST based client application already exists and can or should not be changed.
- Reduce network traffic.
- Limit the type and amount of queries.
- Easier to cache, using GET request.
- Security: Limit access to a well-defined list of queries and effectively prevent access to other contents.

The REST access to persisted queries is enabled by default. If REST access is not desired, the feature can be disabled by adding its autoconfiguration class to SpringBoots exclude list, e.g. in an environment variable.

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

The REST mapping of persisted queries allows for issuing REST requests instead of GraphQL queries. A list of REST endpoints can be configured which map the request to a corresponding [Section 9.7, “Persisted Queries” \[256\]](#). Moreover, the query result can optionally be transformed using JSLT in order to meet the client requirements.

All REST endpoints and their corresponding persisted queries are listed and visualized in the Swagger-UI.

CoreMedia delivers the following examples of persisted queries with the *Headless Server*:

<b>article , page , picture , site</b>	Executes a '... by Id' GraphQL query.
<b>search</b>	Executes a generic 'Search' GraphQL query.

The response of persisted queries using the GraphQL endpoint is JSON as specified by [graphql.org](https://graphql.org). However, it is possible to invoke a JSLT transformation on the result transparently when using the REST endpoint to a persisted query. The files specifying the JSLT transformation must have the same name as the persisted query ID for which they are intended for. These files are stored in the folder `resources/transformations`. In addition to that, it is possible to define a default or fallback transformation by creating a file called `default.jslt` (see next [Section 9.10.2, “JSLT Transformation” \[269\]](#) for details).

The corresponding REST endpoints to the example persisted queries are:

- <https://<your-host>/caas/v1/article/<id>>
- <https://<your-host>/caas/v1/page/<id>>
- <https://<your-host>/caas/v1/picture/<id>>
- <https://<your-host>/caas/v1/site/<siteId>>
- <https://<your-host>/caas/v1/search>

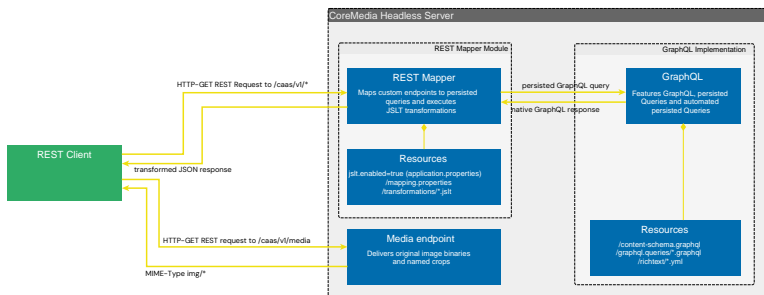


Figure 9.3. Headless server request/response flow using REST

## 9.10.1 Mapping REST Access to Persisted Queries

Every persisted query may be accessed via REST. To enable access via REST, it is necessary to add a mapping of the persisted query to the intended endpoint. By default, the mapping is defined in the file `resources/graphql/rest-mapping/simple-mapping.properties`.

The name of the mapping file is configurable with the property `caas.rest.query-mapping-pattern`. The pattern must be suitable for a Spring `PathMatchingResourcePatternResolver` which is used to load these resources.

```
# Spring PathMatchingResourcePattern to file or files
# Defaults to 'graphql/rest-mapping/*.properties'
# Example:
caas.rest.query-mapping-pattern = graphql/rest-mapping/my-mappings.properties
```

Any persisted query which should be made accessible via REST must be mapped with the filename of the file where the query is defined, without the extension (.graphql), followed by an equal sign and the intended mapping. The mapping

file expects one mapping per line. The format of a mapping may be one of the following two. Both possibilities are equivalent.

```
# 1. query-id = uri-template
article = /article/{id}

# 2. query-id = JSON Object containing at least the key 'uriTemplate' with
the template as its value.
# e.g. article = {"uriTemplate": "/article/{id}"}
```

Additionally, the JSON object must contain type mappings for all query variables, which are not of type "string". Due to the fact that http parameters are strings by nature, the stricter validation of GraphQL query parameters requires a conversion of http parameters into the correct type. Supported conversion types are these basic scalars:

- boolean
- integer
- float

The mapping then may look similar to this:

```
# query-id = JSON Object containing type mapping additionally.
search = {"uriTemplate": "/search", "limit": "integer", "offset": "integer",
"includeSubTypes": "boolean"}
```

The mapping file allows commenting lines via a # in the beginning of a line. Empty lines are also ignored, so using them for grouping is no problem. The mapped URI fragment is always relative to the endpoint `/caas/v1`.

The mapped URI fragment should not end with a slash. However, if it does, the trailing slash will be automatically removed upon system startup.

In addition to a plain URI fragment, it is allowed to add REST path parameters to the mapped URI fragment using the URI template pattern: `{myPathVariable}`. The path parameters are automatically dispatched to the persisted query as GraphQL variables, as well as any query parameters.

## 9.10.2 JSLT Transformation

Depending on the requirements of a REST client, it may be desirable to transform the rather generic GraphQL JSON response into a custom JSON structure. You can do this, using JSLT transformations.

JSLT is a transformation language for JSON, inspired by jq, XPath, and XQuery. For more information and reference about it, please refer to the [JSLT documentation](#).

JSLT transformation templates must be stored in this path: `resources/transformations`. Example transformation templates for all persisted queries are delivered:

```
article.jslt
_default.jslt
errors.jslt
page.jslt
picture.jslt
site.jslt
search.jslt
```

The delivered default transformations are very basic. They simply unwrap the outer two elements of the standard GraphQL response to the pure result data. Furthermore, they showcase how to include a centralized error handling using the JSLT import directive.

A JSLT transformation file is invoked transparently using the name of the invoked persisted query. Whenever a corresponding transformation file is not found, a fallback transformation defined in `default.jslt` is invoked instead, if it exists. CoreMedia provides a fallback transformation template in the file `_default.jslt`, which simply returns the input as the output (= no transformation). To enable this fallback mechanism, rename `_default.jslt` to `default.jslt`. If the fallback template is missing, the JSLT processor is not invoked at all.

Developing more complex transformations may be time consuming as the transformations are read only once when invoked for the first time. Changes on the transformation files only take place after a restart of the *Headless Server*. To overcome this, the online [JSLT evaluator](#) is very useful. Just copy the original GraphQL response to the 'input' textarea and use the 'JSLT' textarea to develop any JSLT transformation and see result directly by clicking the `Run!` button.

## 9.11 Metadata Root

The Metadata Root provides custom metadata for fields. It is configured via a GraphQL schema extension within the file `metadata-schema.graphql` and implemented in the class `MetadataRoot`.

The Metadata Root delivers type definitions retrieved via introspection together with their fields. The fields are enriched with metadata information. The following type definitions are supported:

- `InterfaceTypesDefinition`
- `ObjectTypesDefinition`

Query to retrieve metadata:

```
{
  metadata {
    types {
      name
      fields {
        name
        metadata
      }
    }
  }
}
```

## Customization

Custom metadata can be added by adding a bean of type `MetadataProvider` to the Spring context.

## Configuration

The Metadata Root can be disabled by adding its autoconfiguration class to SpringBoots exclude list, e.g. in an environment variable.

```
SPRING_AUTOCONFIGURE_EXCLUDE=com.coremedia.caas.web.metadata.MetadataAutoConfiguration
```

### 9.11.1 PDE Mapping as Metadata

To integrate PDE (preview driven editing) functionality to a client, a mapping from the field name in the GraphQL schema to the content type property is re-

quired. This mapping is defined on the Headless Server and delivered via `MetadataProvider` as metadata on fields.

## Configuration

The field to property name mapping is configured in file(s) at a configurable location (`classpath*:graphql/metadata/propertyMapping*.json`) as part of Blueprint with a configurable default filename (`propertyMapping.json`), see [Section 3.3.3, "Metadata Properties"](#) in *Deployment Manual* for details.

To add a new custom property mapping file definition, either change the location or the default filename and add the custom property mapping file definition accordingly.

To merge the default property mapping with a custom mapping, add a custom file to the default location and choose a name that matches the given pattern but is different from the default filename, for example, `propertyMapping-custom.json`. The default file is then loaded first, so that subsequent files can override the values.

The entries in the property mapping file consist of interface types that wrap the mapping of field name to the content type property name.

Property mapping configuration (`propertyMapping.json`):

```
{
  "CMCollection": {
    "teasableItems": "properties.items",
    "bannerItems": "properties.items",
    "detailItems": "properties.items"
  },
  ...
}
```

The configured mapping applies also to types that implement the interface.

Configuration is only required for fields whose name differs from the content type property name and for implied content properties.

The default mapping for fields is `"<fieldname>": "properties.<fieldname>"`.

Implied content properties like `id`, `type` etc. are suffixed with `"_"` and need to be configured explicitly in the mapping file. A default configuration is provided in `propertyMapping.json`.

The response of a metadata request containing PDE mapping looks like:

```
{
  "data": {
    "metadata": {
```

```

"types": [
  {
    "name": "CMCollectionImpl",
    "fields": [
      {
        "name": "id",
        "metadata": {
          "mapping": "id_"
        }
      },
      {
        "name": "teasableItems",
        "metadata": {
          "mapping": "properties.items"
        }
      },
      ...
    ]
  }
]

```

## Scope

PDE mapping metadata is provided for `ObjectTypeDefinitions` that implement an interface, for example `CMArticleImpl`. The restriction is applied, because the PDE field mapping is not required for root types and custom object types. The mapping is also not available for `InterfaceTypeDefinitions`, for example `CMArticle`.

The `MetadataProvider` for PDE Mapping is configured for preview only, as PDE is only available in preview apps and typically used to preview data in Studio.

## 9.13 Frontend Client Development

Web apps, created with the React JavaScript library, are a great way to present content from the CMS to consumers via the headless server. This section provides general information and a guide to set up and develop a React app with the Apollo framework. Apollo connects to the GraphQL endpoint of a CoreMedia headless server and fetches the data to display a CoreMedia page, for which Apollo fits best. This setup and its structure are a recommendation to get started quickly and efficiently. Of course other frameworks or different approaches are possible.

### 9.13.1 Rendering the Homepage of a Site

This chapter goes through all necessary steps to render a site's homepage, it's PageGrid and Placements. All starting from the `path` of the page. For cleaner, smaller files, a better overview and to have GraphQL queries separated, this app uses one component for each content item like page or pageGrid etc.

#### 9.13.1.1 Page Component and Query

The page is the entry point for the site and is loading essential data for the homepage like the PageGrid, PageGridPlacements and the banners or collections. So the query in the `Page.jsx` loads this content and passes it down to all other view components. The Query looks like this:

```
const PAGE_QUERY = gql`
  query PageQuery($pagePath: String!) {
    content {
      pageByPath(path: $pagePath) {
        id
        title
        grid {
          rows {
            placements {
              name
              items {
                ... on CMTeasable {
                  id
                  teaserText
                  teaserTitle
                }
              }
            }
          }
        }
      }
    }
  }
`
```



```
},
```

#### Example 9.6. Page query with siteID

The `pagePath` is passed to the `useQuery` hook as an `variables` option, so it is available to the query. The path in our example is `"corporate"`.

From the received data, the rows are now passed on as an array to the `PageGrid` component by applying the spread operator on `data.content.page.grid`. But only if `grid` has any content. To test this it can be written as Boolean equation with the `'&&'` operator, as shown in the example.

The page itself is a good place to start layouting the app, since it is the first component to render to the DOM. So a header and footer component for example could be added here too.

```
function Page(props) {
  const pagePath = "corporate";
  const { loading, error, data } = useQuery(PAGE_QUERY, {
    variables: { pagePath },
  });

  if (loading) {
    return <p>Loading...</p>;
  }
  if (error) {
    return <p>Error :(</p>;
  }

  return (
    <div className="page">
      {data.content.pageByPath.grid && <PageGrid
        {...data.content.pageByPath.grid} />}
    </div>
  );
}
```

#### Example 9.7. Page Component render function

### 9.13.1.2 PageGrid Component

The `PageGrid` component now iterates over the rows and their containing placements, to structure the content into several `PageGridPlacement` components. The key parameter is required by React to have a unique identifier for rendering multiple of the same component at once.

```
function PageGrid(props) {
  const rows = props.rows || [];
  return (
    <>
      {rows.map((row) =>
        row.placements.map(
          (placement) =>
```

```

/>    placement && <PageGridPlacement key={placement.name} {...placement}
    )
  })
</>
};
}

```

*Example 9.8. Iterating over all rows of the PageGrid*

### 9.13.1.3 PageGridPlacement Component

For this example app the resulting web page will look very basic. So for any banner, it renders only the `teaserTitle` and `teaserText`. How to render an image is described in [the following section](#).

```

const divStyle = {
  border: '1px solid black',
  margin: '10px',
  padding: '10px'
};

function PageGridPlacement(props) {
  const name = props.name;
  const items = props.items || [];
  return (
    (items.length > 0 &&
    <div className={name} style={divStyle}>
    <h1>Placement: {name}</h1>
    {items.map((item) => (
      ((item.teaserTitle || item.teaserText) && <div style={divStyle}>
      <h2>{item.teaserTitle}</h2>
      <p>{item.teaserText}</p>
      </div>
    ))}
    </div>
    );
  );
}

```

*Example 9.9. The PageGridPlacement Component*

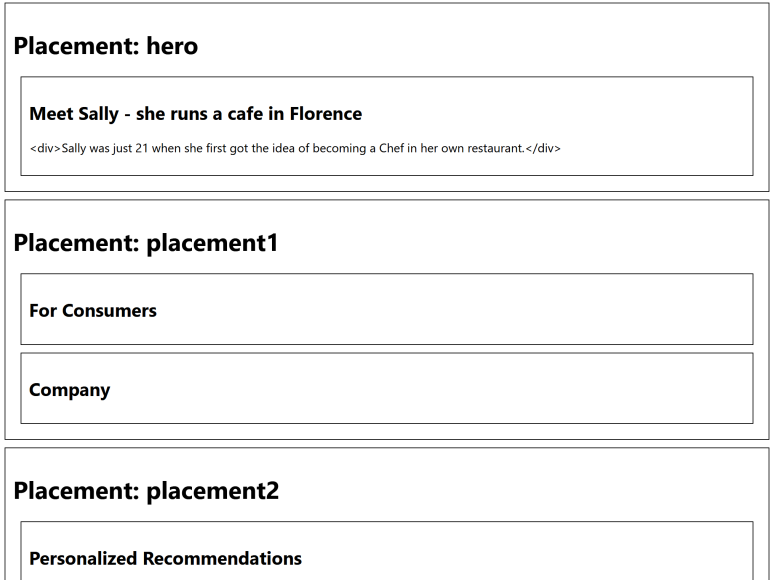


Figure 9.4. Screenshot of the example homepage

**NOTE**

Writing everything in one component can quickly lead to large and messy files. To prevent this, the query can be imported from a separate file in the components folder.



## 10. CoreMedia Campaigns

A campaign is a way to schedule content from CoreMedia Content Cloud to be visible on different, predefined slots on your website for a specific event.

Like advertising slots in the public (for example, main station entrance east 1st large billboard) or website ads, there are named slots available on your website. For example, a "Hero" slot exists for all pages. Campaign content can be shown in these slots. It is, for instance, also possible to create a campaign for a "summer-collection of dresses" only to be shown on category pages below "apparel > woman".

## 10.1 CoreMedia Campaigns Overview

You create a campaign, using the Campaigns user interface, available from CoreMedia Studio. Here you can specify which content from CoreMedia Content Cloud should appear in which slot and on what pages of your website.

### 10.1.1 Visibility of Campaigns

#### Date

To make campaigns run only for specific events, they are scheduled to be active between a start and end date/time. You will plan the active duration for each campaign and CoreMedia Campaigns Delivery API will automatically determine which campaign is currently active at each point in time.

#### Priority

If multiple campaigns run in parallel, a prioritization value on each campaign helps to define an order between those campaigns. For example, a general campaign *"10% discount for Newsletter signup"* should be shown everywhere on the website in the "hero" slot. It has a default priority of 5. If the before mentioned *"summer collection of dresses"* campaign has a priority of 7, it will be shown instead of the medium prioritized *"10% discount for Newsletter signup"* campaign. Of course, you can also decide that you want to display both, for example, in a carousel. In this case, the prioritization is simply used to define the order of content items in a specific slot.

### 10.1.2 Characteristics of Campaign Content

#### Relation to a page

It is necessary to define on which page types your campaign content should be shown. For instance, you can choose a category page. By adding refinements to the page type, you can also narrow down the assignment to only specific sub-pages. For example, only on category page "woman".

### Relation to a slot

In addition to the page, you have to define in what specific slot the content should be shown. When you have defined to show content for a category page, you can now define to shown in the hero slot of these pages.

### Content

Lastly, you choose the content for the slot.

## 10.1.3 What is CoreMedia Campaigns?

With the CoreMedia Campaigns Delivery API the frontend developer gets one endpoint to retrieve content IDs that are managed via the Campaign App for the queried site-id, channel types and refinements. These IDs then can be used to request more detailed information about single content from the Headless Server.

## 10.1.4 Features

Campaigns offers a great interface to request campaign data from. You can either use the builtin [GrapIQI](#) or the [ApolloGraphQL](#) interfaces.

### Restriction by channel type

Campaigns can be defined for different types of pages (for instance: category pages, cms pages and the like). These different types are called "channel type".

### Refinements

In the window of the channel type "Category Page" you can see possible filtering words divided by comma. These filters are called refinements in this context.

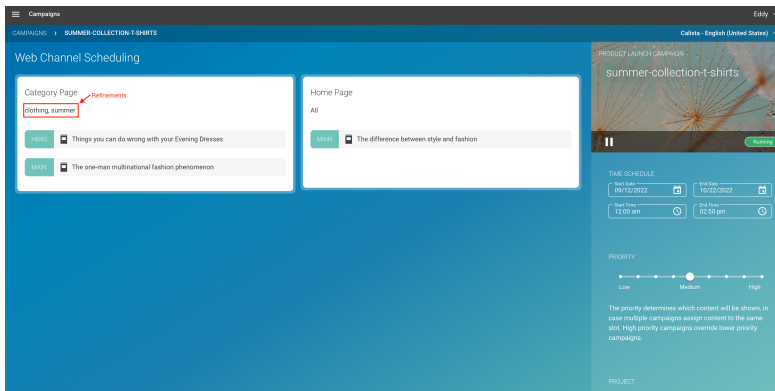


Figure 10.1. Example campaign with defined refinements

## Assignment

The "assignment" is the relation between the following parts: \* A **Content** is \* defined for a **slot** \* inside a **page** \* connected to a **campaign** \* plus additional information provided by **refinements**

To give an example–assignment seen in the picture above:

The **content** *Things you can do wrong with your evening Dresses*, defined for the hero **slot**, inside the category **page**, connected to the **campaign** *summer-collection-t-shirts* and specified by **refinements** *summer* and *clothing*.

## 10.1.5 Getting the Content with GraphQL

As CoreMedia Campaigns is a headless service, it offers a Delivery API based on GraphQL. This API can be used by your website frontend to request the currently active campaign content for a specific page and slot. All campaigns that are currently actively running and that define content for a page and furthermore slot can be returned.

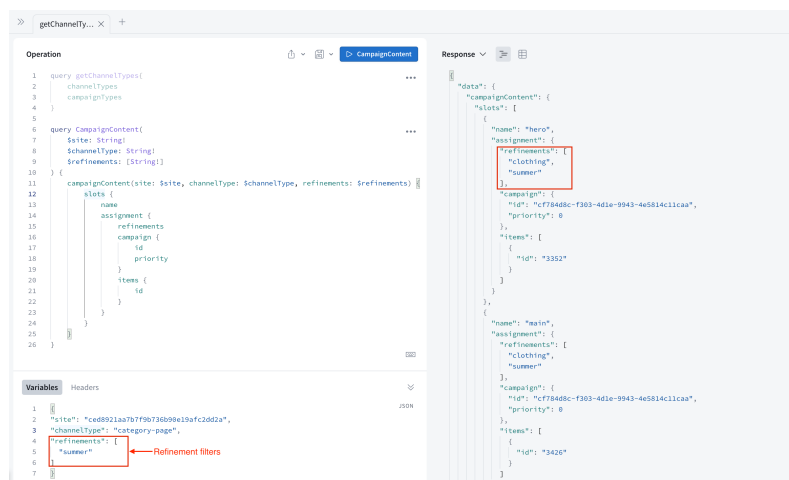


Figure 10.2. Only the summer collection Campaigns are retrieved for the site with the id "ced8921..."

To get the desired content managed via campaigns, for instance, all campaigns that should be visible on category pages and are visible for the summer, a query like shown in the picture above would be correct.

## How do I get a specific campaign?

You don't. The Campaigns Delivery API is not designed to retrieve specific campaigns by name or ID. It is rather the other way around. You perform a request where you specify the site and page you are working on and refinements relevant to that page ("women", for instance) and the API will return the appropriate content from all active campaigns.

## What is the content of a campaign?

The content is divided by the channel types it references to, and after that, a campaign contains linked articles for slots, for example "Hero-slot".

## Which campaign is returned if two with the same refinements exist?

The campaign with the highest priority will be returned if no exact match via refinements can be found. You can also request more than one results via the API, then you get them sorted by priority.



# 10.1.6 Overview of the Architecture

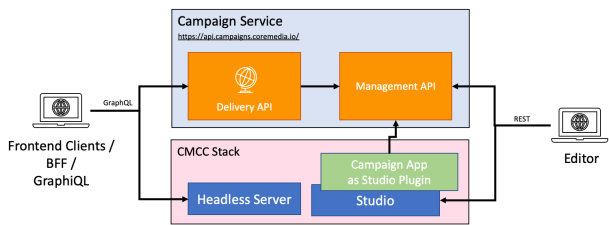


Figure 10.3. Overview of the architecture

# 10.1.7 Campaign Delivery

Let's assume a category landing page (clothing > dresses) is rendered. The client requests the Campaigns Delivery API with the channel type "category-page" and the refinements "clothing" and "dresses". The only campaign that defines content for this page is the "summer-collection dresses" campaign, because it is valid for every category page with refinement *summer* or *clothing*. The Delivery API will return the content UUIDs of the CoreMedia content that was assigned to the "hero" slot in the Campaign App for the Campaign "summer-collection dresses". Of course this only happens within the configured start and end date for the campaign and only if the campaign was set active.

# 10.2 Create a Campaign

## 1. Collecting Content from a Project

Before you can start to create a campaign, you have to collect the content you want to use into a project as described in [Using Projects](#).

Keep in mind that for campaigns, a project is simply used as a container to collect all content needed for the campaign. All other features of a project have no impact on your campaign.

### NOTE

Don't have all the content items in place at the point in time when you're creating the campaign? No problem. You can easily add content to the project at any time.



## 2. Switching to Campaigns

There are two ways to open the Campaign App:

- 1. Click the *Create Campaign* button in the Project window:

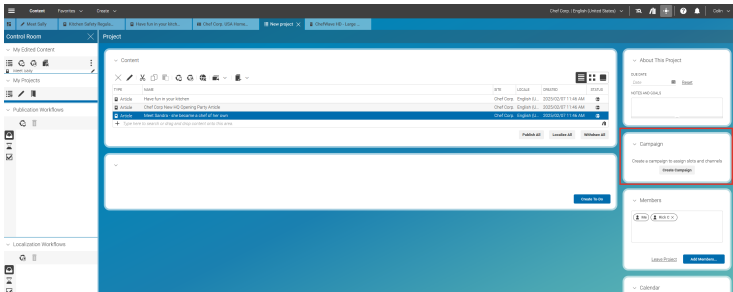


Figure 10.4. Start Campaign Creation from Project

### NOTE

The button 'Create Campaign' will be replaced by 'Open in Campaign App' as soon as the campaign is created.



- 2. Click the *Campaigns* menu item in the main menu of CoreMedia Studio:

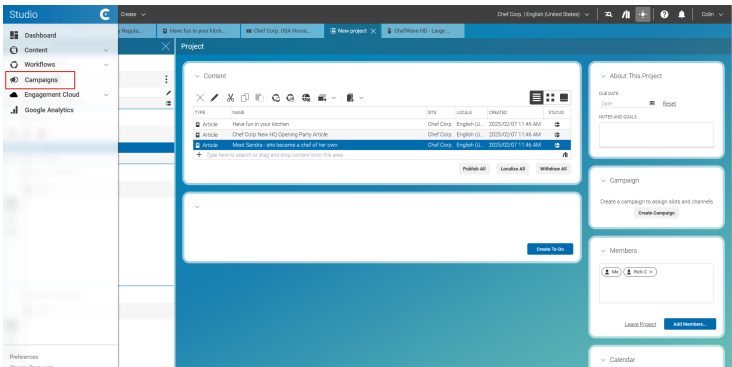


Figure 10.5. Start Campaign Creation from the Main Menu

Once you have opened CoreMedia Campaigns for the first time, you can simply switch between the Content App and Campaigns App by selecting the appropriate tab in your browser.

3. Open Create New Campaign Dialog

When you switch to Campaigns from the project, you will see the main view and the *Create New Campaign* dialog is already opened.

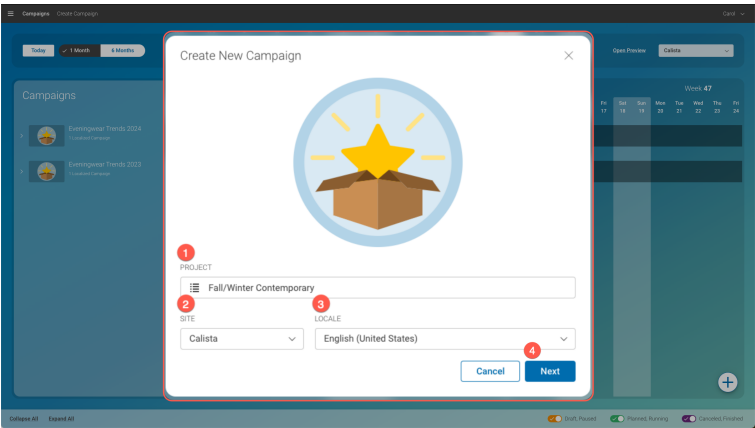


Figure 10.6. Create New Campaigns Dialog

If this is not the case, you can always click the + button to open the *Template Chooser*.

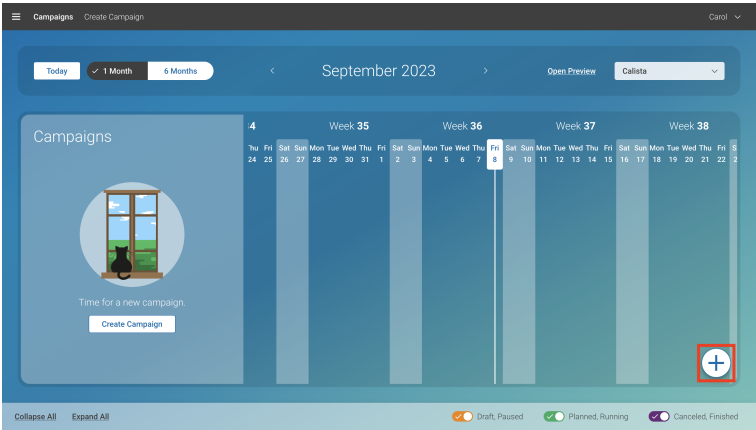


Figure 10.7. Campaigns with Create Campaign Button

4. Enter Campaign Data

Now you can start to define the main data of your campaign.

- 1. Select the project from which you want to take your content. The field is prefilled with your current projects.
- 2. Select the site in which you want to run the campaign. The field is prefilled with your preferred site.
- 3. Select the locale for which you want to create the campaign.

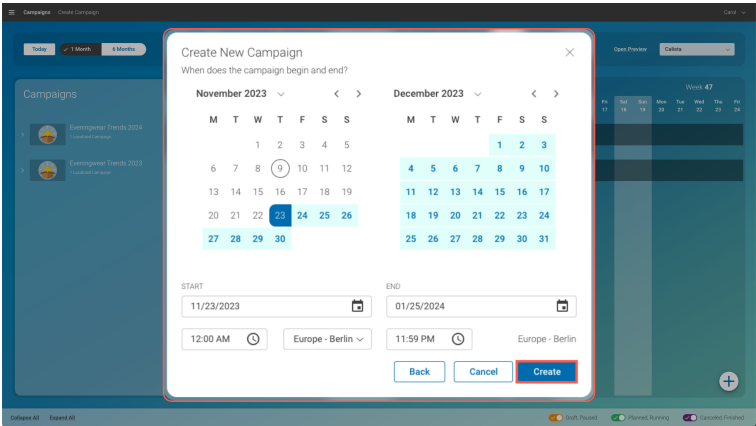


Figure 10.8. Enter start and end date of campaign

Enter the start date and end date of your campaign. Click *Create* to create the Campaign.

### 5. Assign Content to the Campaign Slots

Now, you can define the content to be shown in the slots of the different channels.

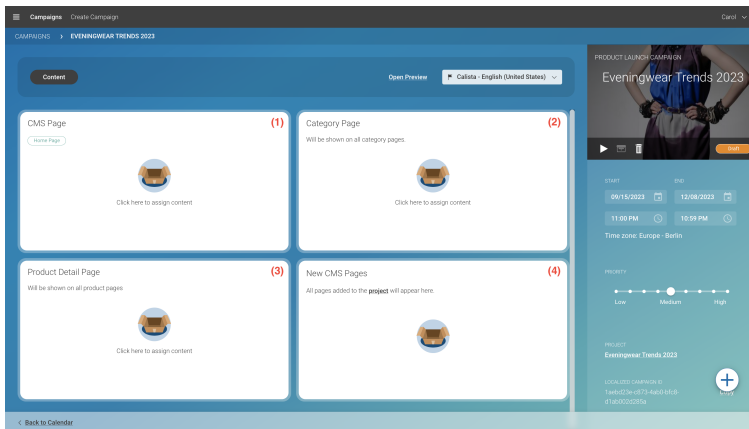


Figure 10.9. Campaign Details

Select the channel for which you want to define the content. In this example, the template contains CMS Pages (1) and Category Pages (2). Under Product Detail Page (3) you can assign content which will be shown on all product pages. Under New CMS Pages (4), all pages that you have added to the project will be displayed. To choose a channel, click on one of the tiles.

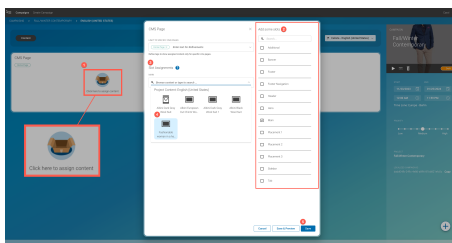


Figure 10.10. Assign content

Define in more detail which content should be shown in which pages and slots:

1. Your CoreMedia installation might contain more than one CMS page. Click into the field *LIMIT TO SPECIFIC CMS PAGES* (1), and you will get a list of all

CMS pages. Mark the checkbox of the pages which should show your campaign content.

2. Each page has several slots in which your campaign content can be shown. Click into the respective slot field (2). You will see a list of all content that you have added to the project. Select one content for each slot and click *Save*.

Proceed in the same way for the other channels of your campaign.

### 6. Prioritize and Start your Campaign

When you are done with all settings and assignments, you can prioritize and start the campaign:

Carol ▾

PRODUCT LAUNCH CAMPAIGN

## Eveningwear Trends 2023

(2) (3)

▶ 📅 🗑️ Draft

START END

09/15/2023 📅 12/08/2023 📅

11:00 PM 🕒 10:59 PM 🕒

Time zone: Europe - Berlin

(1)

PRIORITY

Low Medium High

PROJECT

Eveningwear Trends 2023

Figure 10.11. Prioritize and Start your Campaign

1. When you have more than one campaign running at the same time, they might share the same channels or slots. In this case, you can prioritize campaigns. Content of campaigns with higher prioritization will be favored in comparison to lower prioritized campaigns depending on the actual delivery implementation. Using the priority chooser (1), you can set the priority.
2. When everything is done, click the *Start* button (2) to activate your campaign.
7. Your Campaign is Ready

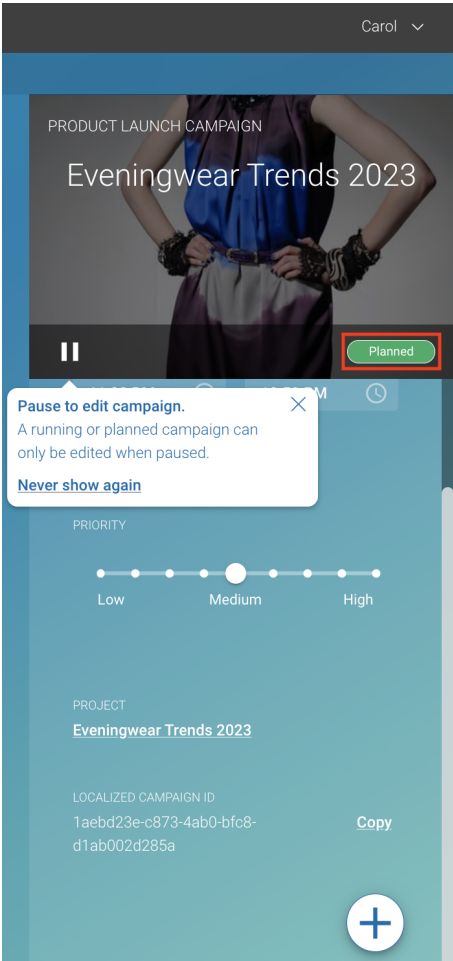


Figure 10.12. Running Campaign

NOTE

In this example, you have chosen a start date in the future. Therefore, once you click the *Start* button, the campaign does not immediately start. Once the date in the future is reached, the status of your campaign will switch from *Planned* to *Running*.





# 11. CoreMedia Event Hub Service

The CoreMedia Event Hub Service is a cloud-only service, which enables you to subscribe to the events of CoreMedia Content Cloud. An event is created whenever a content item has been changed, created, or deleted on either the Content Management Server or the Master Live Server. Additionally, certain user actions in CoreMedia Studio and changes in workflows can trigger events.

## Event Hub Architecture

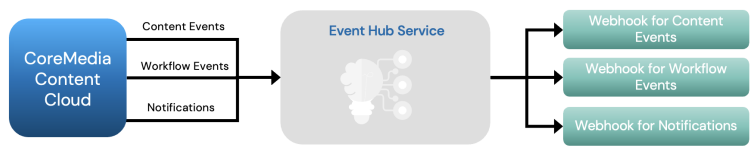


Figure 11.1. Architecture of Event Hub Service

# 11.1 Introduction to the Event Hub Service

Events happening on a CoreMedia Content Cloud instance are sent to the Event Hub Service. The service receives the events, processes, and distributes them to the registered webhooks.

## 11.1.1 Events

You can use the Event Hub Service to consume three kinds of events:

- Content Events

Events from the content repository, analogous to the content events in the CoreMedia Unified API. Examples are:

- content creation
- content update
- check-in/out
- approval
- publication/depublishation
- deletion

- Workflow Events

Workflow relevant events like

- Workflow assignments
- Workflow status changes

- Notification Events

All events that are also available as Studio notifications are available as editorial events in the Event Hub. This includes:

- *Editorial Comments* in Studio
- Other notifications like project assignments.

## 11.1.2 Webhooks

Webhooks are simple URL endpoints. The Event Hub Service sends the event payload in JSON format in the request body to registered webhook endpoints. The API of events is documented in <https://documentation.core-media.com/eventhub-api/>

The webhook endpoint must respond with one of the following status codes:

- OK (200)
- ACCEPTED (202)
- NO\_CONTENT (204)

The endpoint should respond as fast as possible to avoid timeouts on the sender side which would lead to repeated retransmissions of the same message. It's recommended to handle the incoming messages asynchronously. That is, persisting them to a durable storage and returning the webhook call and using asynchronous workers to do the real processing on the persisted messages.

The content events are delivered in batches of up to 100 items.

There will be some delay between the change on the Content Server and the arrival at the webhook endpoint. The delay depends on the system load and can be up to 1 minute.

### Error Cases

The events are delivered sequential in the FIFO (first in, first out) manner. That has an implication when the webhook endpoint responds to a message with something else than the 2xx status codes listed above: The message will be repeatedly sent until the endpoint responds with a 2xx. Until then, no later messages will be sent to the webhook endpoint. Though, the retry is limited by our internal timeout of 7 days. After reaching the timeout, the unsuccessfully responded message will be removed from the system.

### Configuration of the Webhook Subscription

You can use the Event Hub Webhook Subscription Service to administer the webhook endpoints for your CMOC instances by yourself. See [Using the Event Hub Webhook Subscription Service](#)

### Activation of the Event Hub Service

Contact the CoreMedia support at [support@coremediaoncloud.com](mailto:support@coremediaoncloud.com) to get the Event Hub Service activated for your account.

The CoreMedia support will provide you with the URL of the subscription service and will provide secrets in the Cloud Manager of your instance. Use the secrets to get the token, necessary to access the subscription service.

## 11.2 Working with the Event Hub Service

The CoreMedia Event Hub Service is a cloud-only service, which enables you to subscribe to the events of CoreMedia Content Cloud. See [Introduction to the Event Hub Service](#) for how to activate the service for your instance. This how-to document shows you how you can administer webhook endpoints by yourself, using the event subscription service. The API of the subscription service is documented in <https://documentation.coremedia.com/eventdeliveryconfig-api/>.

### 11.2.1 Prerequisites

- The URL of the webhook subscription service, which is provided by the CoreMedia support
- The access token as the service is protected by JWT. The CoreMedia support showed you how to access the token. All calls to the subscription service must include the header Authorization Bearer <JWT token>
- The 8-character tenant ID of your CMOC instance, also provided by the CoreMedia support.
- The environment (live or preview) from which you want to consume the events.

### 11.2.2 Create New Webhook Subscription

You register a new webhook using the call `POST /<environment>.<tenant id>/config`

The configuration of the webhook endpoint is sent as JSON body in the request, for example:

```
{
  "description": "Customer ACME, sandbox first",
  "namespace": "PUBLIC",
  "subscriptionTypes": [
    "CONTENT",
    "NOTIFICATION",
    "WORKFLOW"
  ]
  "enabled": true,
  "endpoint": {
    "url": "https://acme.com/firstsandbox/eventHubHook",
    "method": "POST",
```

```
"authenticationMethod": {
  "authenticationType": "BASIC_AUTH",
  "username": "evenhub-first-sandbox-user",
  "password": "passwordForEventHubHook"
},
"headerParameters": {
  "Authorization": "Bearer 5bbryflnm897q38j6a9ug9p5rfehf5mcbn7"
}
}
```

- Provide a description for the configured webhook.
- The namespace must be PUBLIC.
- The subscriptionTypes are one or more from the three different event types the webhook can consume:
  - CONTENT
  - NOTIFICATION
  - WORKFLOW

For the live environment only the CONTENT event type is available.

- enabled must be true to activate this webhook. Later you can set it to false to deactivate the webhook without deleting the webhook.

The endpoint configures the actual communication from the event hub service to the webhook:

- url is the URL of the webhook endpoint. Only https URLs are allowed.
- The request method must be POST or PUT.
- authenticationMethod configures the authentication method used by the event hub service when sending the events to the webhook URL. Currently, only basic authentication is supported. To that end, set authenticationType to BASIC\_AUTH and username and password. Contact CoreMedia for other authentication schemes.
- Alternatively you can use a bearer authentication setting a header in headerParameters, which are the list of header parameters which are sent to the webhook URL along with the events. The example below shows an example of Authorization header parameter.

```
{
  "description": "Customer ACME, sandbox first",
  "namespace": "PUBLIC",
  "subscriptionTypes": [
    "CONTENT",
    "NOTIFICATION",
    "WORKFLOW"
  ]
  "enabled": true,
  "endpoint": {
```

```
"url": "https://acme.com/firstsandbox/eventHubHook",
"method": "POST",
"headerParameters": {
  "Authorization": "Bearer 5bbryflnm897q38j6a9ug9p5rfehf5mcbn7"
}
}
```

If successful, the response body looks like this:

```
{
  "items": [
    {
      "id": "a324392a-8b69-4e1e-a72b-22a02d1291f8",
      "tenantId": "&lt;tenant id&gt;",
      "sourceId": "preview",
      "description": "Customer ACME, sandbox first",
      "namespace": "PUBLIC",
      "subscriptionTypes": [
        "CONTENT",
        "NOTIFICATION",
        "WORKFLOW"
      ],
      "enabled": true,
      "endpoint": {
        "endpointType": "WEBHOOK",
        "url": "https://acme.com/firstsandbox/eventHubHook",
        "method": "POST",
        "authenticationMethod": {
          "authenticationType": "BASIC_AUTH",
          "username": "xxxxxxx",
          "password": "xxxxxxx"
        }
      }
    }
  ]
}
```

Sensitive data like password, user and headers are hidden in the response.

You can configure multiple subscriptions for the same event type, but there are some restrictions as shown below.

## Return Code

A successful creation of the webhook subscription returns 201 (Created). You will get 400 (Bad request) if the configuration is invalid, for example, an invalid webhook URL. In some cases, the subscription service will decline the subscription and return 409 (Conflict):

- Duplicate endpoint: You are trying to subscribe the same webhook endpoint which has been already subscribed to the same event type.
- Maximum number of subscriptions: For a given pair of environment and tenant there is a limit of subscriptions, which is currently 10.
- Maximum number of subscriptions for an event type: For a given triple of environment, tenant and event type there is a limit of subscriptions, which is currently 4.

In case of an unsuccessful request, the response body contains JSON with an error message. For example,

```
{
  "statusCode": 400,
  "messages": [
    "endpoint.url: must be a valid URL"
  ]
}
```

See [Section 11.2.6, “Table of Error Codes and Messages” \[300\]](#) for the comprehensive list of all error codes and messages.

## 11.2.3 Read Existing Webhook Subscriptions

You get all registered webhook subscriptions for a given environment and tenant using the request `GET /<environment>.<tenant id>/config`

The response body looks like below

```
{
  "items": [
    {
      "id": "a324392a-8b69-4e1e-a72b-22a02d1291f8",
      "tenantId": "<tenant id>",
      "sourceId": "preview",
      "description": "Customer ACME, sandbox first",
      "namespace": "PUBLIC",
      "subscriptionTypes": [
        "CONTENT"
      ],
      "enabled": true,
      "endpoint": {
        "endpointType": "WEBHOOK",
        "url": "https://acme.com/firstsandbox/eventHubHookForContentEvents",
        "method": "POST",
        "authenticationMethod": {
          "authenticationType": "BASIC_AUTH",
          "username": "xxxxxxx",
          "password": "xxxxxxx"
        }
      },
    },
    {
      "id": "36e7d0cd-b8d4-42f8-89e8-e1af4b990e70",
      "tenantId": "<tenant id>",
      "sourceId": "preview",
      "description": "Customer ACME, sandbox first",
      "namespace": "PUBLIC",
      "subscriptionTypes": [
        "NOTIFICATION",
        "WORKFLOW"
      ],
      "enabled": true,
      "endpoint": {
        "endpointType": "WEBHOOK",
        "url": "https://acme.com/firstsandbox/eventHubHookForNotificatonAndWorkflows",
        "method": "POST",
        "headerParameters": {
          "Authorization": "xxxxxxx"
        }
      },
    }
  ]
}
```



```

}
}
}

}
}

```

## 11.2.4 Update Existing Webhook Subscriptions

The example above shows two subscriptions for the tenant and environment preview, one with the ID a324392a-8b69-4e1e-a72b-22a02d1291f8 and one with 36e7d0cd-b8d4-42f8-89e8-e1af4b990e70. You can change the subscription using the following call: **PUT /<environment>.<tenant id>/config/<subscription id>**

As for the creation of a new webhook subscription, send the changed configuration as a JSON body. For example, to disable the subscription with the ID a324392a-8b69-4e1e-a72b-22a02d1291f8 send the call **PUT /preview.<tenant id>/config/a324392a-8b69-4e1e-a72b-22a02d1291f8** with the following JSON body:

```

{
  "description": "Customer ACME, sandbox first",
  "namespace": "PUBLIC",
  "subscriptionTypes": [
    "CONTENT"
  ],
  "enabled": false,
  "endpoint": {
    "endpointType": "WEBHOOK",
    "url": "https://acme.com/firstsandbox/eventHubHookForContentEvents",
    "method": "POST",
    "authenticationMethod": {
      "authenticationType": "BASIC_AUTH",
      "username": "eventhub-first-sandbox-user",
      "password": "passwordForEventHubHook"
    }
  }
}

```

## 11.2.5 Delete Existing Webhook Subscription

Delete an existing webhook subscription using the call **DELETE /<environment>.<tenant id>/config/<subscription id>**

For example, to delete the subscription with the ID a324392a-8b69-4e1e-a72b-22a02d1291f8 send the following call:

DELETE /preview.<tenant id>/config/a324392a-8b69-4e1e-a72b-22a02d1291f8

## 11.2.6 Table of Error Codes and Messages

Return Code	Message	Description
400 (Bad Request)	Missing input	The required JSON body is missing.
	Invalid ID	The ID given for GET/PUT/DELETE requests on individual subscriptions is not a valid UUID.
	Input cannot be deserialized	The given request body is no valid JSON according to the spec. This can be generally malformed input or, for example, misspelled enum values.
	Namespace '<name>' is forbidden	The calling user doesn't have sufficient rights to use the requested subscription namespace.
	Namespace 'PUBLIC' does not allow endpoint type '<endpoint_type>'	Only webhook endpoints are allowed for the PUBLIC namespace.
	Namespace 'PUBLIC' does not allow subscription of '<event_type>' events	The PUBLIC namespace only allows the subscription to CONTENT, NOTIFICATION and WORKFLOW events.
	Use of service name is reserved	A subscription with 'PUBLIC' namespace must not define a service name.
	Service name is required	Only relevant for internal subscriptions. An internal subscription must have a service name.
	Namespace is immutable	The namespace assigned when creating a subscription cannot be changed when updating the subscription.

Return Code	Message	Description
	<property path> <message>	A property within the submitted JSON has a validation error, that is, a null value where a value is required.
409 (Conflict)	Duplicate endpoint '<end-point_url>'	A specific endpoint can only be used once per event type.
	Maximum number of 10 subscriptions reached	The total number of subscriptions is limited to 10.
	Maximum number of 4 subscriptions for stream '<stream_type>' reached	The total number of subscriptions per event type is limited to 4.

Table 11.1. Table of error codes and Messages

# 12. Image Transformation Service

*CoreMedia Content Cloud – Service* customers can benefit from an image transformation based on a shared service that supports the WebP format out-of-the-box. Due to the nature of shared services, it has different features and limitations compared to the built-in image transformation library used in earlier versions and in self-managed installations of the *CoreMedia Content Cloud*. These differences are described in the following sections.

This information applies to new *CoreMedia Content Cloud – Service* installations after September 2023.

## Sizes, Formats and Operations

The image transformation feature of *CoreMedia Content Cloud – Service* supports the following images in terms of file sizes (*MB* = Megabytes, *MP* = Megapixels), file formats and transformation operations:

Input Dimensions	
Max. image file size	20 MB
Max. image megapixels	25 MP
Max. multi-frame file size	40 MB
Max. multi-frame megapixels	200 MP
Output Dimensions	
General – <i>Applies if no other limit is defined for the format</i>	25 MP (e.g., 5000 x 5000 pixels)
Animated GIF	120 MP (e.g., 3446 x 3446 with 10 frames or 2000 x 2000 with 30 frames)
Animated WebP	200 MP (e.g., 4472 x 4472 with 10 frames or 2581 x 2581 with 30 frames)

Operations

All operations known from the built-in transformation are supported except the rotation of animated formats.

Formats

Upload	JPEG, PNG, GIF
Transform	WebP, AVIF, JPEG, PNG, GIF

Table 12.1. Sizes, formats and operations

NOTE

CoreMedia Studio won't prevent the editor from uploading bigger pictures and from creating transformations that do not comply to the constraints above. In these cases, the preview will show a broken image. CoreMedia plans to provide the editor with more detailed feedback in the future.



Continuous Updates

During the lifetime the shared services used for the image transformation will be continuously improved, provided with bugfixes and security updates. As a customer, CoreMedia takes care of these changes for you.

Limited Customizability

The image transformation features are the same for all service customers. They cannot be changed for individual customers in the shared services. Customizations of image delivery in the CAE and the Headless Server are consequently only possible to a certain degree.

Getting Access As An Existing Customer

If you are a *CoreMedia Content Cloud – Service* customer using the built-in Image Transformation library in the CAE and the Headless Server, contact the CoreMedia support at [support@coremediaoncloud.com](mailto:support@coremediaoncloud.com) to learn about the prerequisites that enable you to use the cloud-only image transformation.

# 13. CoreMedia Ingest Service

The CoreMedia Ingest Service is a cloud only service enabling you to develop your own importer-like applications for the CoreMedia Content Cloud.

It offers an API with basic methods to create and modify content items. In addition, the content lifecycle is supported with bulk publication, unpublication and deletion methods (see [Ingest Service API Reference](#)).

This how-to document introduces you to the following tasks:

- Learn how to get the Ingest Service
- Learn the basic steps to import content, see [section “Performing a Basic Content Import” \[305\]](#).
- Learn about different ways to import blob data, see [section “Handling Binary Objects” \[305\]](#).
- Learn how to use asynchronous calls for bulk operations such as publication or deletion of content, see [section “Bulk Repository Operations” \[306\]](#).
- Learn how to use concurrent calls to improve content creation speed and how to size content sets for bulk operations, see [section “Increasing Throughput” \[308\]](#).
- Learn about the OpenAPI UI for testing and its limitation, see [section “OpenAPI UI for Testing and Limitations” \[308\]](#).

The CoreMedia Ingest Service is located behind a gateway which limits the maximum response time of an API call to 30 seconds by default. If this limit is exceeded, clients will receive a *Service Unavailable (503)* response from the gateway even though the request may successfully finish on the service side.

There are two types of API calls that are prone to request timeouts, running synchronous bulk operations (see [Bulk Operations](#) and using *UriBlobProperty* models for very large (in the range of some hundred megabytes) binary objects (see [section “Handling Binary Objects” \[305\]](#)).

## Getting the Ingest Service

Contact the CoreMedia support at [support@coremediaoncloud.com](mailto:support@coremediaoncloud.com) to get the Ingest Service activated for your account.

## Performing a Basic Content Import

Figure Figure 13.1, “Import Sequence” [388] shows a schematic sequence of API calls for importing a single content to CoreMedia (handling of binary data left out, see section “Handling Binary Objects” [305]).

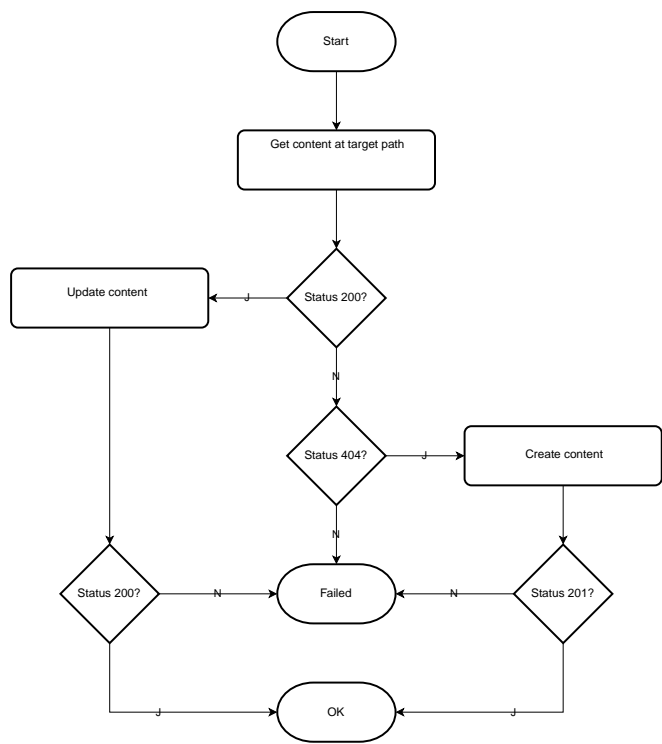


Figure 13.1. Import Sequence

Additional checks, like verifying the content type before trying to update a content, could be added. If the target location is guaranteed to be free of already existing content, which is the case when importing to a new subfolder structure for the first time, the initial check for content at the target path can be omitted.

## Handling Binary Objects

There are currently three different ways to import CoreMedia content items with blob properties:

- *createTempBlob*: The recommended way for productive use.
- *UrlBlobProperty*: For an easier client, but not recommended for productive use.
- *postBlobData*: For an update of an existing blob property.

### *createTempBlob* (recommended)

Upload the binary data upfront by creating a *Temporary Server Side BLOB* (*createTempBlob*). Then use the property model from the call's response in the following API call to create or update the content. This is the recommended way to handle binary objects, since it avoids request timeouts and leaves the client in complete control of handling the data source.

### *UrlBlobProperty*

Upload the binary data using a *UrlBlobProperty* model.

Passing only a URL for the data source, the service will have to read the binary data when creating the document.

This method has drawbacks:

- Depending on the size of the binary data, the requests are prone to timeouts. Depending on the available network throughput this can take some time. For the gateway it will appear as if the service is not available, and it will terminate the request with a 503 response.
- The data source is limited to HTTP(S) or S3 URLs which must be accessible from the service. This also means, the source cannot be protected by any sophisticated authentication mechanism.

### *postBlobData*

Update a single BLOB property after a content has been created (*postBlobData*).

This is only recommended for existing content if and only if a single blob has to be updated. Otherwise, there would be two or more transactions involved:

- First create or update the content with all non blob properties.
- Afterwards update every single blob in the content in a separate transaction.

This is not only slower than the other methods, it also creates superfluous intermediate content versions with possibly invalid state.

## Bulk Repository Operations

Bulk repository operations can be executed in two modes:

- synchronous
- asynchronous (recommended)



Using synchronous mode may at first seem easier from a client perspective, but is prone to request timeouts if either the number of content items for the bulk operation is too large, or the publisher queue is overloaded with too many requests.

**Asynchronous bulk repository operations are therefore recommended.**

Figure Figure 13.2, “Sequence for asynchronous bulk requests” [307] shows a simplified schematic sequence for bulk operations.

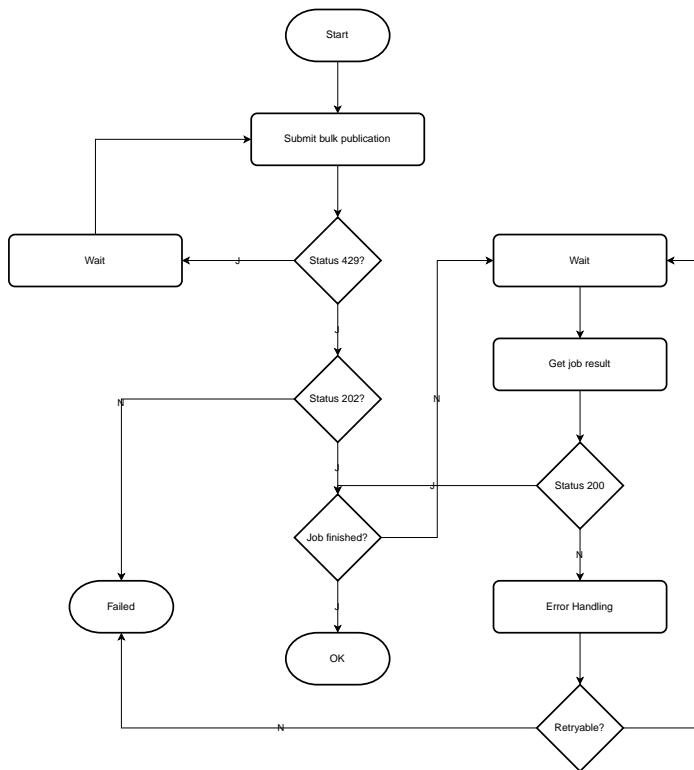


Figure 13.2. Sequence for asynchronous bulk requests

When doing bulk operations, keep the following points in mind:

- Synchronous and asynchronous bulk operations may be rejected with a response of *Too Many Requests* (429) if the service’s worker or queue limits are reached. When you get this response, let your client wait some time and retry.

- A response code of 200 does not mean a bulk operation completed successfully (synchronous) or has finished (asynchronous). You always have to examine the state contained in the response object.
- Bulk operations on folders are executed on every transitive child of the folder. So always know the number of children and avoid the operation when the number is too high.
- You do not need to run a *withdraw* operation before a *delete*. Unlike in Studio a *delete* operation will automatically try to withdraw all content items before moving them to the recycle bin.

## Increasing Throughput

In order to optimize the throughput you have to distinct between different operations:

- Single-item operations like content creation and updates
- Bulk operations like publication and deletion of content

### Single-item operations

Content creation and updates are single item operations, which you can run concurrently in order to increase throughput.

Practical experience has shown 10–15 concurrent requests to be an upper limit after which scaling is greatly reduced. Also keep in mind not to overload the system if it is used for productive work.

### Bulk operations

Bulk repository operations, that is publish, unpublish, and delete, don't profit similarly from concurrent invocations. The first phase of a bulk operation, like setting the content approval flags, can run concurrently with other bulk operations, but the final publications are executed strictly sequential.

For bulk operations you can influence the throughput with the size of the passed-in content set. The recommended range lies between 100 and 200 content items. Larger sets do not improve the speed anymore but block the publisher for longer. They can also improve the risk of network problems for synchronous calls (which are not recommended, see [Bulk operations](#)).

## OpenAPI UI for Testing and Limitations

On cloud environments with the Ingest Service available, logged in administrators and developers can access the OpenAPI UI to check out the online documentation and try out the API.

For the following API calls, which use direct data streaming, the *Try Out* mode will not work:

- Create a temporary BLOB (*createTempBlob*)
- Update data of a content's BLOB property (*postBlobData*)

# 14. Integration: CoreMedia Hubs and Connectors

CoreMedia is also introducing Hubs and Connectors, which are all about extending CoreMedia Content Cloud and integrating it with 3rd party services.

Each Hub provides dedicated integration points to connect external systems. It has its own dedicated, stable API that is responsible for the communication between CoreMedia Content Cloud and a 3rd party through a specific Adapter.

A Connector is an integration solution that enables a range of integration use cases. In that sense it serves as a bridge between CoreMedia Content Cloud and 3rd party services. A connector can consist of one or more Hub Adapters and any number of additional extensions to provide the required integration capabilities.

CoreMedia will provide standard Connectors as part of the product or on CoreMedia Labs under the CoreMedia Open Source license. Custom connectors can be built by our partners and customers by using the new Hub APIs and our existing Blueprint extension mechanism. We encourage our partners to build reusable Connectors and to make them available to our joint customers.

Some examples of Connectors and use cases that use multiple Hubs and extension points follow:

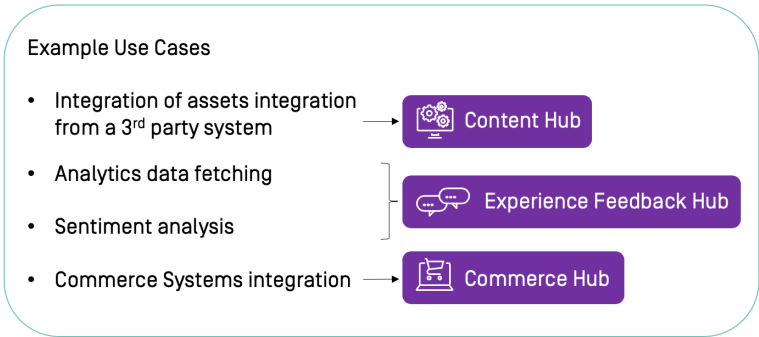


Figure 14.1. Use Cases

One external system can be connected to several Hubs:

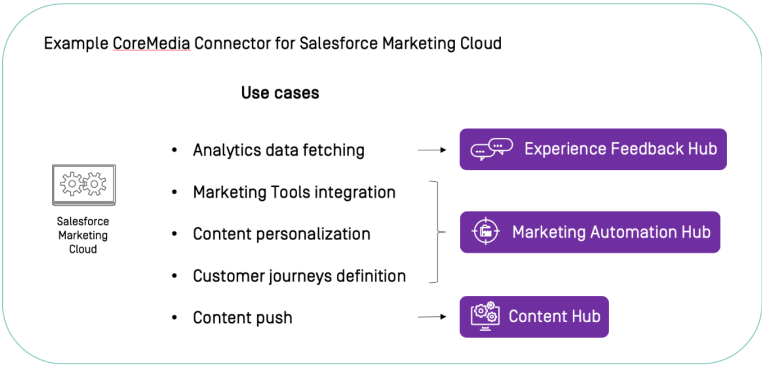


Figure 14.2. Connector for Marketing Cloud

# 14.1 CoreMedia Content Hub

The Content Hub provides an API that enables the integration of external sources into the library of CoreMedia Studio. You can browse the sources and transfer external content into the content repository. Out of the box there are example adapters for RSS Feeds and YouTube. Adapter configuration can be changed during runtime. In the near future, CoreMedia will add preview and search capabilities, duplicate detection, and other important features.

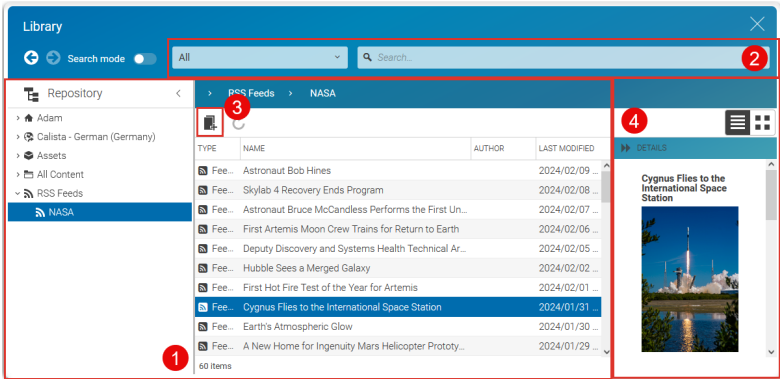


Figure 14.3. Import content from RSS feed

## 14.2 CoreMedia Commerce Hub

The connections to external commerce systems now follow a new architectural approach (called Commerce Hub). A generic, commerce-vendor neutral client in the CMS components (for example, Studio, CAEs) connects to one or more commerce adapter services that contain vendor-specific code. This allows a clear separation of the commerce connection and other CoreMedia components. It enables you to:

- use different commerce providers simultaneously
- avoid huge number of requests and the resulting slow latency towards Commerce systems by adding a delegating and caching service in between, so that not every component directly accesses the commerce system.
- bugfixes and updates for new Commerce releases only need an update of a small component/service independent of the main CoreMedia components

As all connectors to the Commerce systems have different requirements in terms of technology and used libraries, they are now independent of each other and do not share the same libraries in the blueprint, anymore.

Nevertheless, the different adapters and the CMS blueprint use the same base-client that define the API between the generic client and the adapter implementations. The communication protocol uses gPRC with protocol buffers, based on HTTP/2 to ensure a fast and reliable connection.

Demo Sites for SAP Hybris Apparel, Salesforce Commerce Cloud SiteGenesis and SFRA as well a HCL (formerly IBM) Commerce Aurora B2C now use Commerce Hub by default.

Calista still uses the legacy non-Commerce Hub approach and will be migrated in the near future. Legacy eCommerce API implementations for Salesforce Commerce and SAP Hybris have been removed from the blueprint but can be requested via Support.

The following connectors are available:

- Connector for Salesforce Commerce Cloud, see [Section 3.1, "CoreMedia Content Cloud for Salesforce Commerce Cloud"](#) [9]
- Connector for HCL / IBM Commerce, see ???
- Connector for SAP Commerce Cloud, see [Section 3.2, "CoreMedia Content Cloud for SAP Hybris"](#) [31]

# 14.3 CoreMedia Feedback Hub

The Feedback Hub is the place where editors find additional information from other editors, external services and internal services about content. The initial release gathers feedback from the well-known content validators, but there is also an example implementation of an adapter of the new Keywords API. The adapter that implements Imagga's Image Recognition API suggests keywords to pictures in the CoreMedia content repository. In the future, there will be more Feedback Hub adapters and APIs that allow embedding even more information.

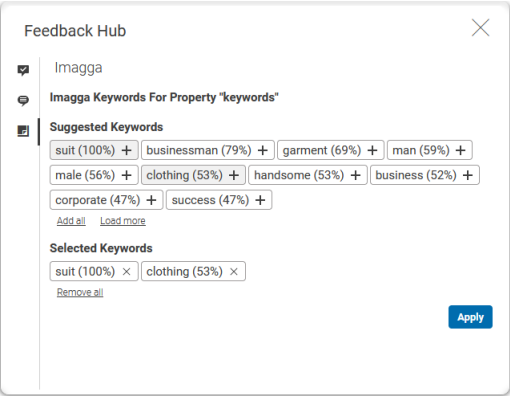


Figure 14.4. Example of Imagga Integration



# 15. CoreMedia Blueprint 3rd party Integrations

CoreMedia Blueprint comes with default integrations of third-party software.

# 15.1 Salesforce Marketing Cloud Integration

Salesforce Marketing Cloud (SFMC) is a customer relationship management (CRM) tool by Salesforce. CoreMedia Content Cloud offers you an integration with the following features:

- The Integration with Salesforce Marketing Cloud enables editorial user to push content from the CoreMedia CMS to Salesforce Marketing Cloud and for example reuse it for campaigns or Newsletters.
- There are also new CoreMedia Personalization rules that leverage SFMC journeys and enables you target CMS content based on the SFMC journey a website visitor is part of.
- Additionally, we provide an API to pass data to SFMC Data Extensions for e.g. storing form data or manage subscriptions.

## Uploading Content to Salesforce Marketing Cloud

CoreMedia Blueprint allows you to upload content to Salesforce Marketing Cloud (SFMC). For configured string, richtext or image blob properties of a given content type CoreMedia Studio has an upload button on the Action toolbar.

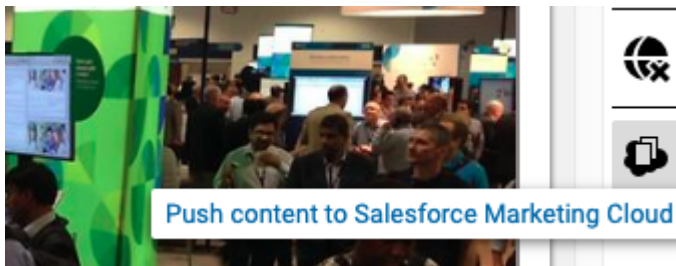


Figure 15.1. Push content to Marketing Cloud

For each uploaded CoreMedia content there is a folder with a name composed of the document name and content ID. In the folder for each uploaded content property there are one or more assets. For an image blob property there is an asset for each of the transformed image crops and the original image. The name of an asset is composed of the content name, property name and – in case of images – the crop name. The uploaded assets can now be used in SFMC, for example as content in a newsletter.

## SFMC Journey Integration

The condition checks whether the current user is associated to a certain SFMC journey. For the configuration of the feature read the section 'Upload Content to Salesforce Marketing Cloud' of the blueprint developer manual.

In the second input field you select a comparison operator from the drop-down box. You can choose if the user "is" or "is not" associated to the journey that you select in the third input field.

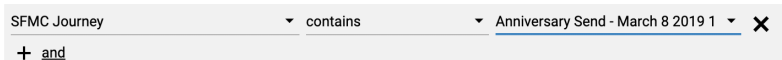


Figure 15.2. SFMC Journey condition example

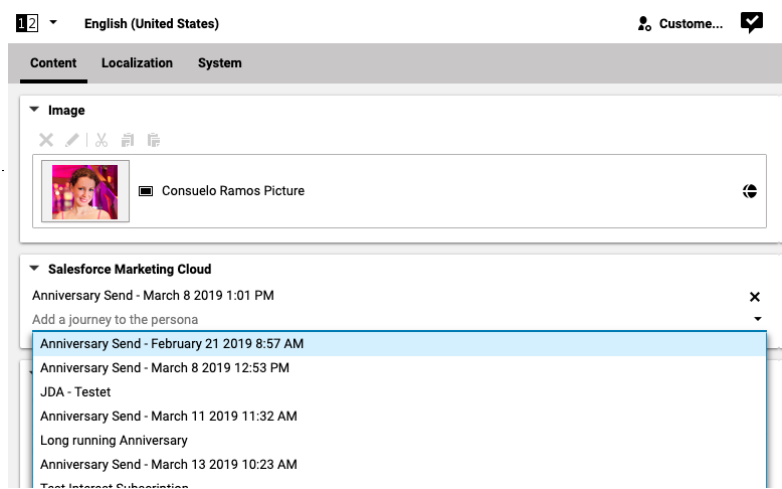


Figure 15.3. Adding SFMC Journeys in personalization preview personas

## API to pass data to SFMC Data Extensions

CoreMedia Content Cloud provides a Service API to push data into SFMC data extensions.

# 15.2 Open Street Map Integration

The Open Street Map project creates and distributes free geographic data. *CoreMedia Blueprint* is prepared to include the project to display the location of location based taxonomies, but map integration are not included in the default templates.

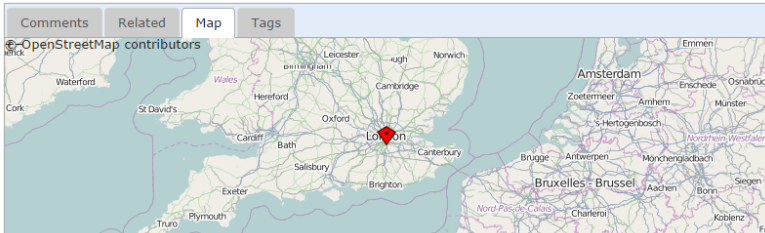


Figure 15.4. Example for an Open Street Map integration in a website

In order to use Open Street Map on your site, you have to create a settings content item and link it to the root channel of your site. The JavaScript for Open Street Map will be loaded using an aspect that is only enabled if the corresponding settings property is set. The available settings for Open Street Map are shown in the table below and must be configured to enable the map in the CAE. A template renders a map segment according to geographic coordinates stored in the string property `latitudeLongitude` of a linked location content, and pinpoints the matching location with a marker image (see `CMTearable.map.ftl` for a usage example).

## 16. CoreMedia Labs



CoreMedia Labs is first and foremost intended to open a dialogue with CoreMedia's customers regarding new and upcoming features: We hope that you will try out these features and tell us how you use them.

Labs is also for developers who want to have a look under the hood or get some hands-on understanding of the vast and compelling capabilities of CoreMedia. Whatever your experience level with CoreMedia is, we've got something for you.

Each project in CoreMedia's Labs platform is an extra feature to be used with CoreMedia, including extensions, tools and 3rd party integrations. CoreMedia provides some test data and explanatory videos for non-customers and for insiders there is open-source code and instructions on integrating the feature into your CoreMedia workspace.

The code CoreMedia provides is meant to be example code, illustrating a set of features that could be used to enhance your CoreMedia experience. We'd love to hear your feedback on use-cases and further developments! Send us an email at [labs@coremedia.com](mailto:labs@coremedia.com).

## 16.1 Feedback Hub Adapter – AB/n Testing

Marketers, listen up. Our goal is to achieve the highest possible conversion rates on our website, right? So we know a good user experience is crucial. We also know that relevant content improves the user experience, and that by better appealing to our target audience we can increase conversions.

But are you grappling with these content questions? For example: How do I write the best title or description? How long should the text be? Which images will attract attention? What would be a good call-to-action? And where should I place it?

It's not easy to answer these questions from my site visitors, since I get no direct feedback from them. And what I really need is to know is their behavior and preferences in detail. Which I'm not going to get from a crystal ball (tempting though that may be).

Fortunately there are solid techniques for gathering qualitative and quantitative insights of user behavior. Analyzing these insights helps me understand the preferences of my target audience better and make data-driven decisions around content.

One of these techniques, of course, is A/B testing. Comparing two versions of a page – variant A versus variant B – can help improve conversion rate and reduce bounce rate. And there are also tools that allow you to compare more than two versions of a page, called A/B/n testing. (Note that this is different from multivariate testing, which allows you to analyze all variation items separately. But this approach requires a high volume of traffic in order to yield statistically significant results.)

The good news for CoreMedia customers is that Studio now enables marketers to easily perform A/B/n split testing from right within the platform – no need to add commercial third-party services. The process is simple: Create at least two variants of the content (aka a baseline and a competing candidate) that differ from each other in ways like text length, structure or image. Then fire up the A/B/n test to determine which variant resonates best with visitors.

Let's take a closer look at this with a simple teaser. A merchandiser for an online shop, for example, wants to figure out where place a product on a landing page so as many customers as possible click on it (and consequently buy it). But those questions again: How to illustrate the product best? And what should the wording of the CTA be?

The solution, of course, is to create two variants of the teaser to see which one resonates best with the target audience. But we can't simply place one next the other on the same page. That would only lead to confusion.

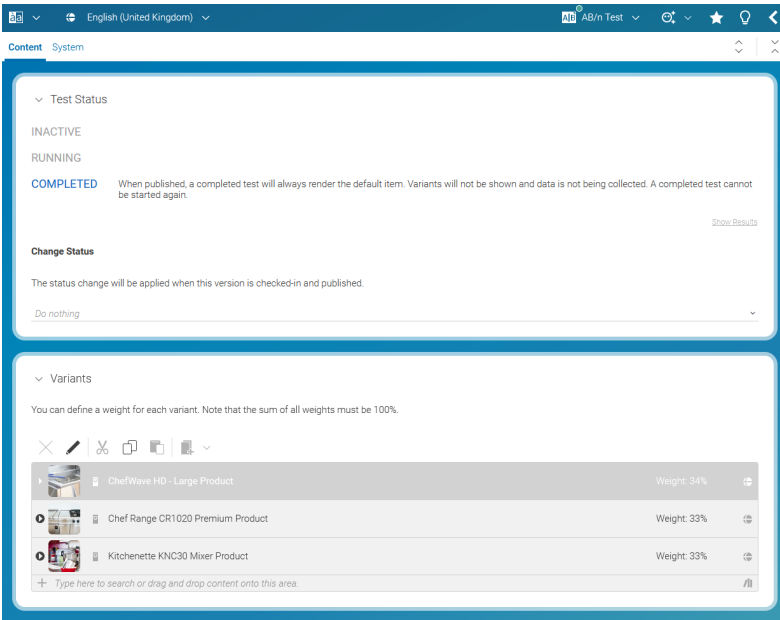


Figure 16.1. Teaser for A/B testing

So in order to compare these two versions and determine a winner, we create a new A/B/n test in CoreMedia Studio and insert both teaser versions as variants. For each variant, we define a “weight” that determines how what percentage of the visitors will see it.

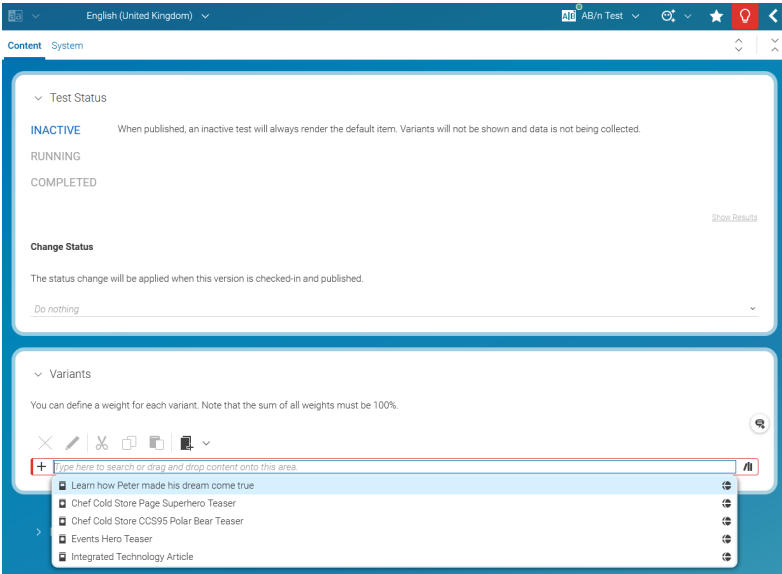


Figure 16.2. Create A/B/n test in Studio

While this experiment runs, each user sees one of the two variants. And this is key: If a user visits the page repeatedly they're always offered the same variant because their session information is stored in a cookie.

Using Google Analytics, we can track the behavior of all website visitors with relevant insights displayed right in CoreMedia Feedback Hub. This is where I can see how many users were shown each variant and how many of them selected it, resulting in a conversion rate comparison to help determine a winner. (Plus, more detailed information about the user behavior can be viewed on the Google Analytics Dashboard).



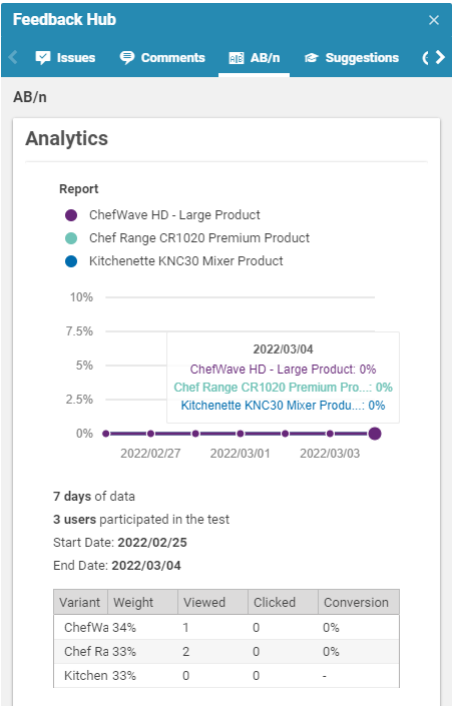


Figure 16.3. A/B/n test results

The CoreMedia A/B/n test add-on has just been released via CoreMedia Labs and is available for all customers with a CoreMedia personalization license – available right out-of-the-box without the need to add a personalization solution or integrate any third-party software. So get it while it's hot.

## 16.2 Feedback Hub Adapter – Acrolinx

This is an integration for the SEO and content marketing platform Acrolinx (<https://www.acrolinx.com/>). The Feedback Hub Adapter 'Acrolinx' is implemented as a Blueprint extension.

### Project Setup

For configuration details see <https://github.com/CoreMedia/feedback-hub-adapter-acrolinx/blob/master/documentation/Configuration.md>.

Acrolinx in the Feedback Hub:

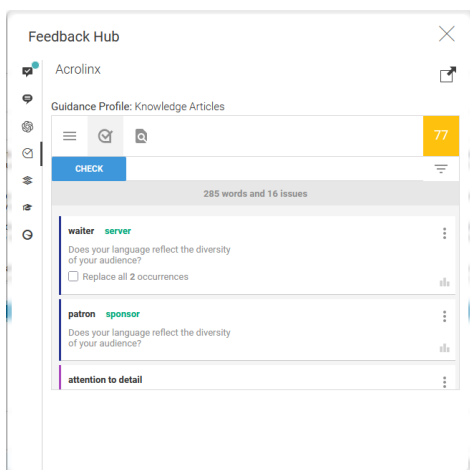


Figure 16.4. Acrolinx window

Feedback for CoreMedia richtext

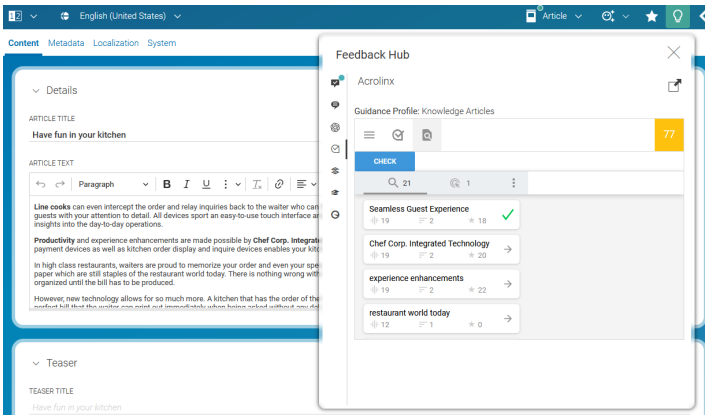


Figure 16.5. Feedback for CoreMedia richtext

# 16.3 Feedback Hub Adapter – Searchmetrics

The *CoreMedia Feedback Hub* offers the possibility to provide feedback for CoreMedia content. It is possible to connect external systems to Feedback Hub in order to collect feedback.

This is an integration for the SEO and content marketing platform *Searchmetrics*.

The Searchmetrics Feedback Hub Adapter integrates Searchmetrics briefings into the CoreMedia Feedback Hub.

## Overall Scoring Overview

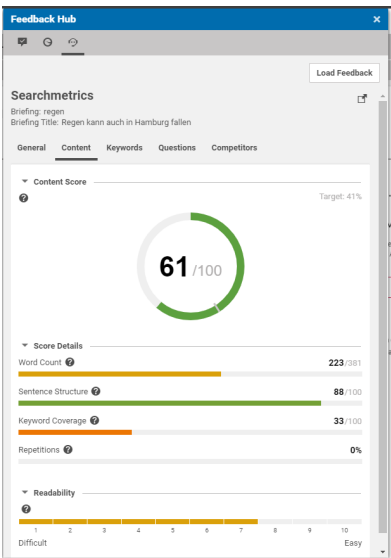


Figure 16.6. Overall Scoring Overview

Keyword Scoring

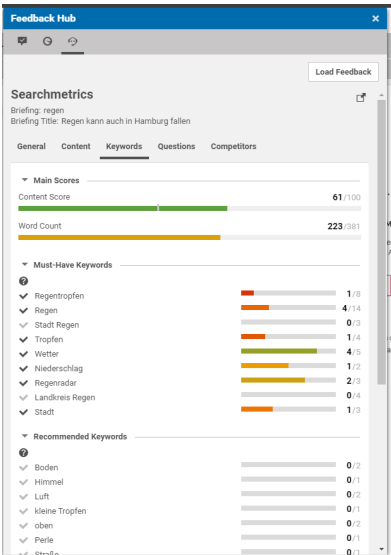


Figure 16.7. Keyword Scoring

Content Questions

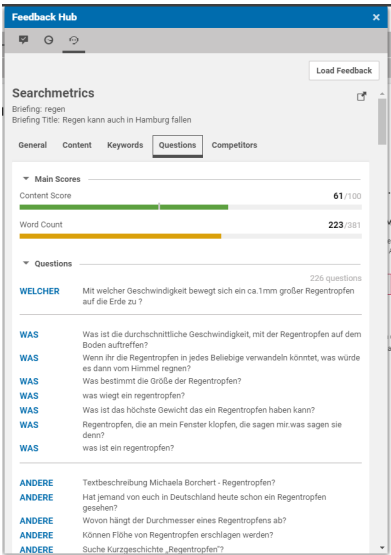


Figure 16.8. Content Questions

Competitors Overview

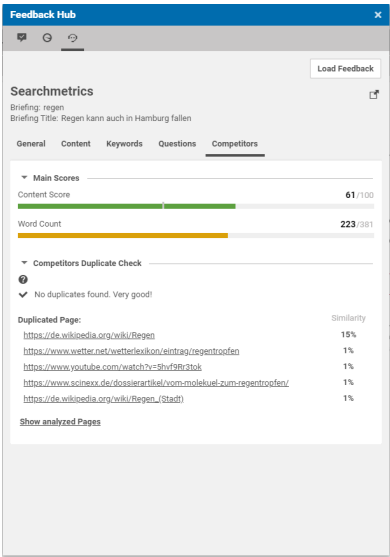


Figure 16.9. Competitors Overview

For more details, see here: <https://github.com/CoreMedia/feedback-hub-adapter-searchmetrics>

## 16.4 Feedback Hub Adapter – Siteimprove

The *CoreMedia Feedback Hub* offers the possibility to provide feedback for CoreMedia content. It is possible to connect external systems to Feedback Hub in order to collect feedback.

This project integrates the *Siteimprove* REST API into the Feedback Hub of CoreMedia. It also enables the Siteimprove widget for the preview CAE.

### Prerequisites

In order to use Siteimprove with CoreMedia, you should already have setup your sites in Siteimprove. Ensure that not only the live CAE is crawled by Siteimprove, but also the preview CAE.

The following screenshot shows the setup of two test sites: Labs Preview and Labs Live.

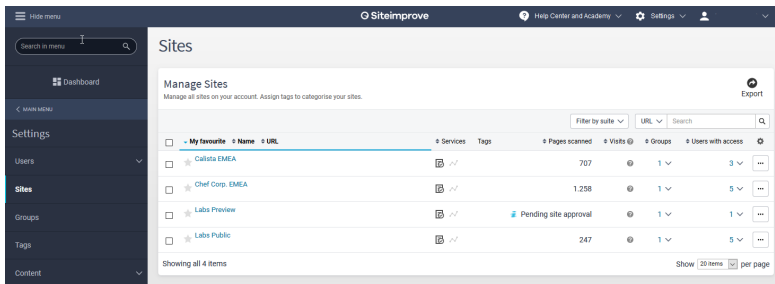


Figure 16.10. Siteimprove Site Setup

### Configuration for the CoreMedia Feedback Hub

The CoreMedia Feedback Hub integration for Siteimprove can be enabled by creating a new global CMSettings content item in a global or site-specific folders.

- **Global:** /Settings/Options/Settings/Feedback Hub/
- **Site specific:** <SITE>/Options/Settings/Feedback Hub/



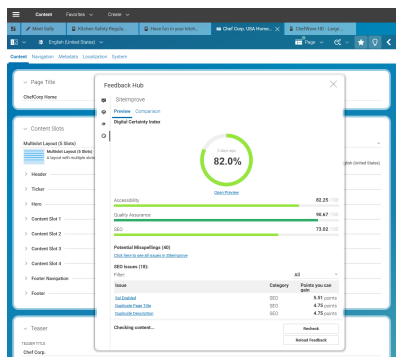


Figure 16.11. Siteimprove Feedback shown in Studio

For more details, see here: <https://github.com/CoreMedia/feedback-hub-adapter-siteimprove>

## 16.5 Feedback Hub Adapter – OpenAI ChatGPT

OpenAI has released several AI-driven services such as ChatGPT to the public. The chatbot is based on GPT-3.5, a variety of deep learning language models. Since then, the whole topic of AI continues to gain attention. Find the CoreMedia OpenAI ChatGPT Connector in the [CoreMedia Marketplace](#).

The reasons are obvious – ChatGPT and GPT are revolutionizing how businesses interact with their customers and how businesses interact internally. With its natural language processing capabilities, ChatGPT can understand human conversations and then generate accurate personalized responses in real-time.

And there's more; AI-driven image generation services like DALL-E, also powered by OpenAI, provide the means to theoretically generate any image that businesses may need to best complement messages with their customers.

CoreMedia has developed an integration with OpenAI's GPT service – again proving the simple composability: With just a little development effort based on the proven CoreMedia Feedback Hub, we now have a seamlessly integrated powerful AI solution that makes a difference.

See an animated image screencast below. It demonstrates the use case: Let GPT generate text based on a question or a general idea.

Entering text like “Write an article on summer fashion trends in a language suited for generation z” will yield the desired text that is at least a great start for your own story.

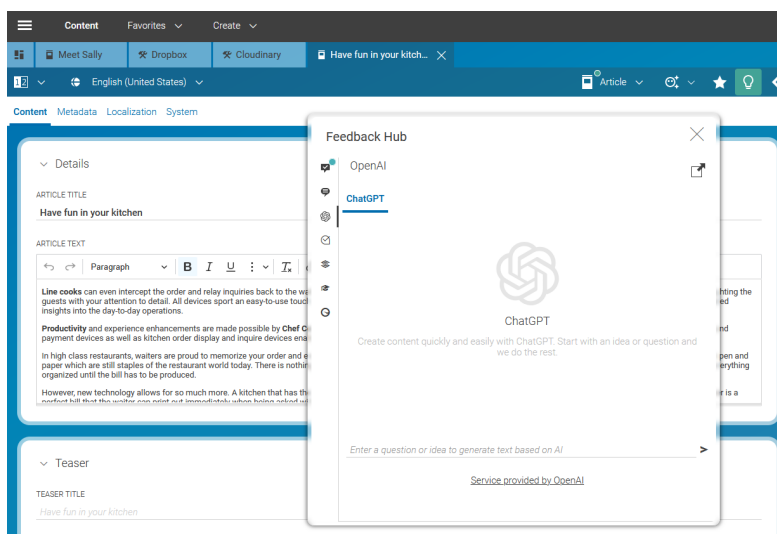


Figure 16.12. ChatGPT Integration in Studio

### Prerequisites

- Feedback Hub
- OpenAI Account

# 16.6 Feedback Hub Adapter – wonk.ai

wonk.ai provides AI-based content services to generate, summarize, or translate text based on custom-trained AI models. Find the CoreMedia wonk.ai Connector in the [CoreMedia Marketplace](#).

With Write, wonk.ai provides an AI-based text generation service. Enter a keyword, limit which sources should be used – and then create unique content in the desired target language also based on current knowledge from data sources in the internet. This is much more than you are used to get with standard ChatGPT like services as Write does not only have their natural language capabilities, but also take into account up-to-date information from the internet – and list the information about the sources used for text generation. This creates a huge difference in terms of text accuracy and quality.

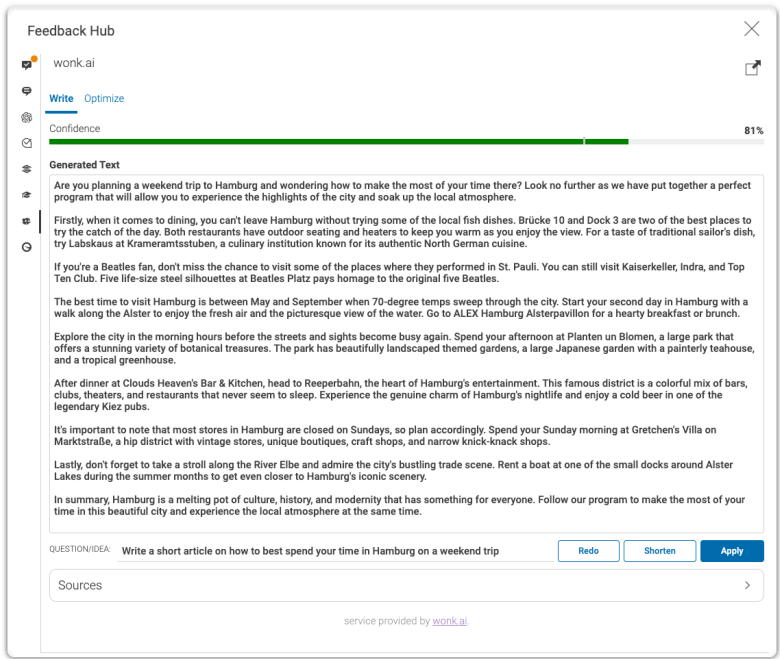


Figure 16.13. Write integration

The wonk.ai Optimize service supports the editor with the creation of search optimized content by generating relevant content:

- To extract keywords or tags from your text
- To generate catchy taglines
- To generate HTML metadata or SERP texts (as shown in every Google search result)

The following screenshots shows Optimize in action:

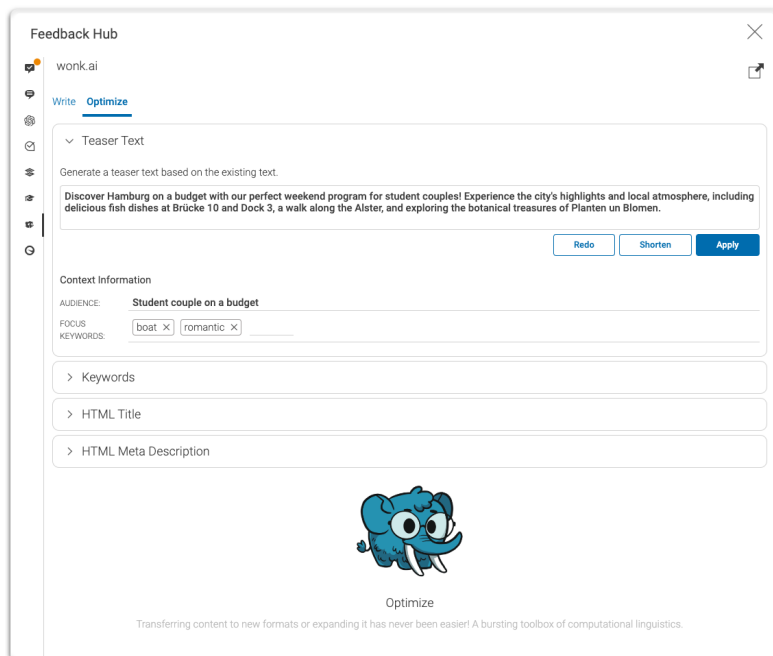


Figure 16.14. Optimize teaser text

All of this is used to optimize metadata mainly for SEO.

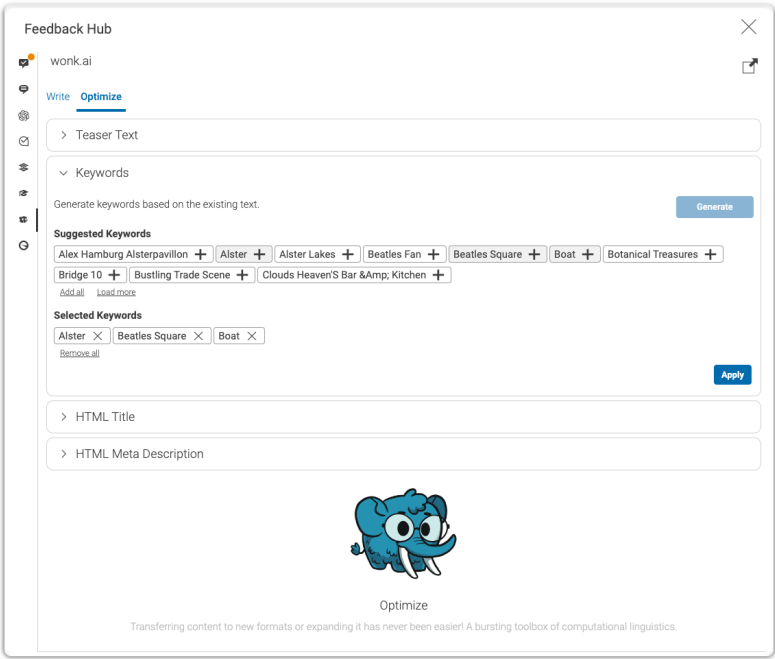


Figure 16.15. Optimize Keywords

**Prerequisites**

- Feedback Hub
- wonk.ai Account

## 16.7 CoreMedia Content Hub Adapter – Dropbox

The *coremedia-content-hub-dropbox* plugin provides access to a configurable Dropbox account. Common to all content-hub-adapters is the appearance in CoreMedia Studio. The image below shows multiple configured content hub adapters in Studio (content-hub-adapter-dropbox is marked with a blue background).

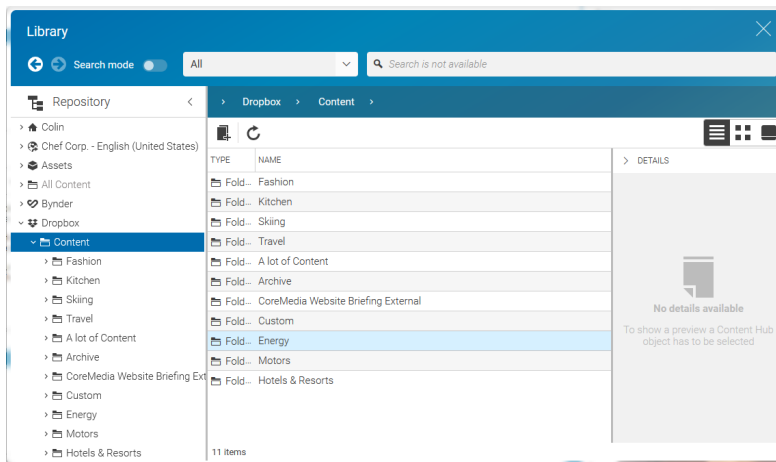


Figure 16.16. Content Hub adapters in Library

### Supported content types for this adapter

- Images
- Audio files
- Videos
- PDFs

# 16.7.1 Configuring the content-hub-adapter-dropbox

Depending on the configuration of the adapter, the appearance of the tree may vary. The following sections take care of all configuration places and options.

## Basic Adapter Configuration

This section is covering the two possibilities to enable the content-hub-adapter-dropbox integration. Please note that those options are valid for all content-hub-adapters. Before configuring the adapter, please refer to the documentation [Content hub configuration](#) for preliminary steps.

### Global Adapter Configuration

To enable the content-hub-adapter-dropbox for all sites, create a CMSettings content item inside the /Settings/Options/Settings/Content Hub/Connections/ folder.

For convenience reasons, it is recommended to name this content item "Dropbox".

### Site Specific Adapter Configuration

To enable the content-hub-adapter-dropbox for a single site, create a CMSettings content item inside the Options/Settings/Content Hub/Connections/ folder, relative to the site's root folder.

For convenience reasons, it is recommended to name this content item "Dropbox".

## Detailed Adapter Configuration

### Basic Structure

The table below shows the initial toplevel entry for all content-hub-connector configurations.

Key	Type	Required
connections	StructList	Yes

Table 16.1. Toplevel entry

After creation of the initial struct list called *connections* the next step is to create the first entry. This can be done in Studio with the struct editor by pressing "add



Item to ListProperty". The table below shows the entries which are common for all connectors.

Key	Type	Value	Required
connectionId	String	<SOME_UNIQUE_ID>	Yes
factoryId	String	<YOUR_FACTORY_ID>	Yes
enabled	Boolean	true or false	Yes
settings	Struct		Yes

Table 16.2. Entries for all connectors

Required Configuration

In section *Basic Structure* above and according to the table, the settings struct is currently empty. The settings struct itself holds specific configuration options for the connector (common to all connectors). The table below shows all potential entries.

Key	Type	Value	Required
accessToken	String	Access token to open a connection to Dropbox	Yes
displayName	String	Name of the root folder to display in Studio	No

Table 16.3. Entries in settings struct

Example

The image below depicts a full configuration of the *content-hub-adapter-dropbox* in global space.

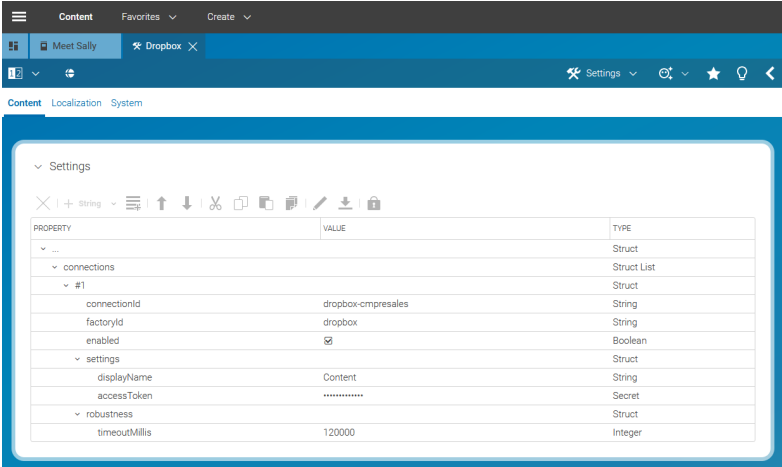


Figure 16.17. Full configuration of adapter in global space

## 16.7.2 Usage

Once the connector is configured, the *Dropbox* named tree appears in the library, and by clicking on *Dropbox* the tree expands and is showing content from Dropbox in the well known folder–content structure from CoreMedia. The image below shows the appearance.

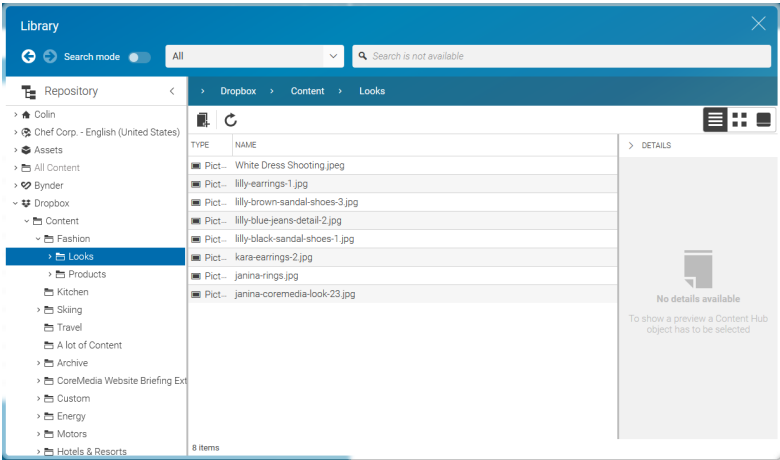


Figure 16.18. Configured Dropbox in Studio Library

By browsing the tree, the content-hub-adapter-dropbox provides an import mechanism for content objects. The picture below shows the button for creation (purple frame).

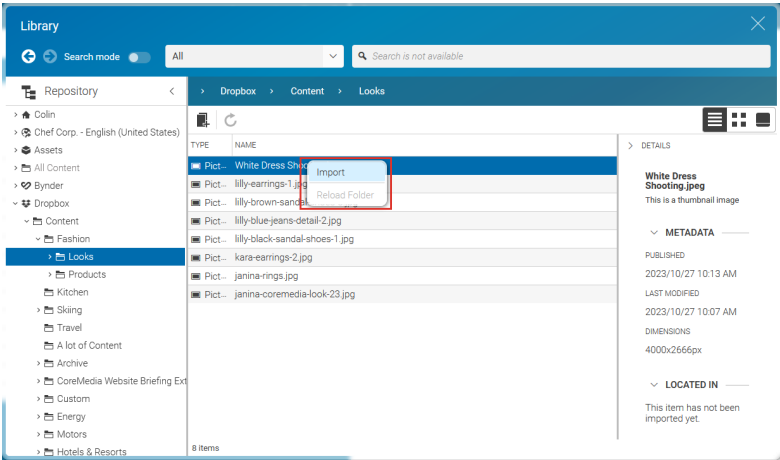


Figure 16.19. Importing content from Dropbox in Studio Library

## 16.8 CoreMedia Content Hub Adapter – Cloudinary

The *coremedia-content-hub-cloudinary* plugin provides access to a Cloudinary account. Common to all Content Hub adapters is the appearance in *CoreMedia Studio*. The image below shows multiple configured Content Hub adapters in *Studio* with the *content-hub-adapter-cloudinary* selected.

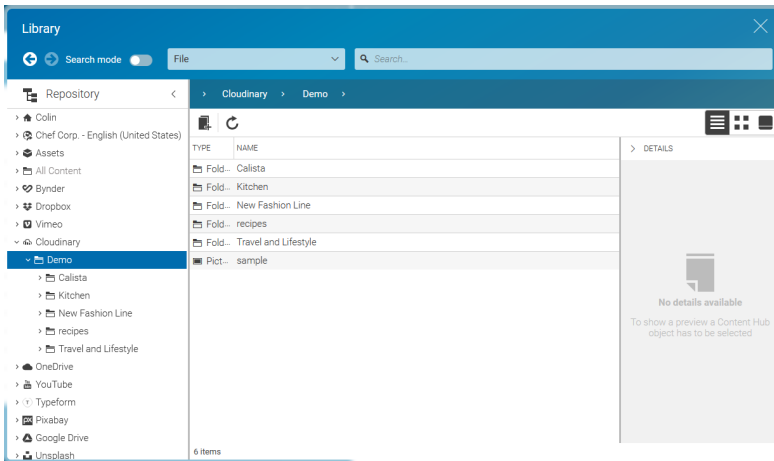


Figure 16.20. Cloudinary adapter in the Library

### Supported content types for this adapter

- Images
- Videos

### 16.8.1 Configuring the content-hub-adapter-cloudinary

Depending on the configuration of the adapter, the appearance of the tree may vary. The following sections take care of all configuration places and options.

## Basic Adapter Configuration

This section covers the two possibilities to enable the *content-hub-adapter-cloudinary* adapter. Please note that those options are valid for all Content Hub adapters. Before configuring the adapter, refer to the documentation [Content hub configuration](#) for preliminary steps.

### Global Adapter Configuration

To enable the *content-hub-adapter-cloudinary* for all sites, create a CMSettings content item inside the `/Settings/Options/Settings/Content Hub/Connections/` folder.

For convenience reasons, it is recommended to name this content item "Cloudinary".

### Site Specific Adapter Configuration

To enable the *content-hub-adapter-cloudinary* for a single site, create a CMSettings content item inside the `Options/Settings/Content Hub/Connections/` folder, relative to the site's root folder.

For convenience reasons, it is recommended to name this content item "Cloudinary".

## Detailed Adapter Configuration

### Basic Structure

The table below shows the initial toplevel entry for all content-hub-connector configurations.

Key	Type	Required
connections	StructList	Yes

Table 16.4. Toplevel entry

After creation of the initial struct list called *connections* the next step is to create the first entry. This can be done in *Studio* with the struct editor by pressing *add Item to ListProperty*". The table below shows the entries which are common for all connectors.

Key	Type	Value	Required
connectionId	String	<SOME_UNIQUE_ID>	Yes
factoryId	String	<YOUR_FACTORY_ID>	Yes

Key	Type	Value	Required
enabled	Boolean	true or false	Yes
settings	Struct		Yes

Table 16.5. Entries for all connectors

Required Configuration

In section *Basic Structure* above and according to the table, the settings struct is currently empty. The settings struct itself holds specific configuration options for the connector (common to all connectors). The table below shows all potential entries.

Key	Type	Value	Required
displayName	String	Name of the root folder to display in Studio	No
cloudName	String	Account name to open a connection to Cloudinary	Yes
apiKey	String	API key corresponding to the Cloudinary account	Yes
apiSecret	String	API secret corresponding to the Cloudinary account	Yes

Table 16.6. Entries in settings struct

Example

The image below depicts a full configuration of the *content-hub-adapter-cloudinary* in global space.

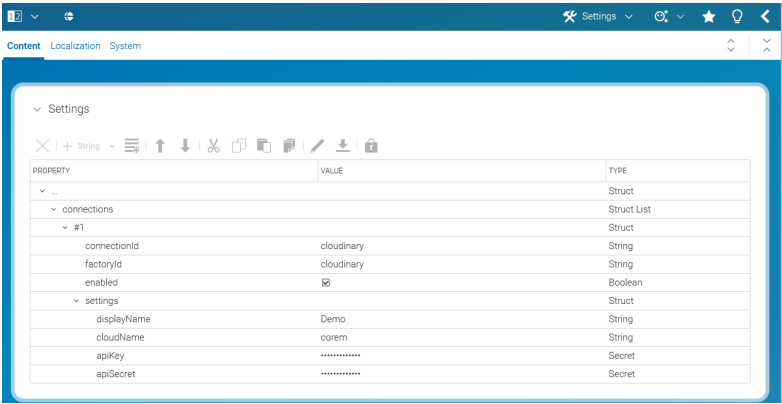


Figure 16.21. Full configuration of adapter in global space

## 16.8.2 Usage

Once the connector is configured, the *Cloudinary* named tree appears in the library, and by clicking on *Cloudinary* the tree expands and is showing content from the file system in the well known folder-content structure from CoreMedia. The image below shows the appearance.

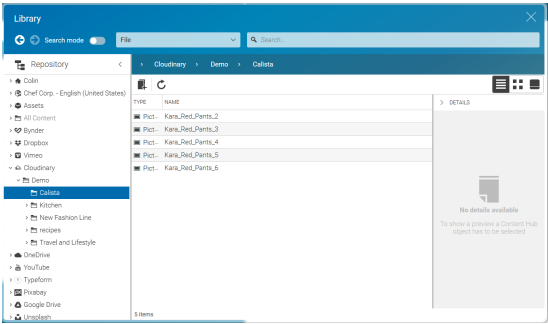


Figure 16.22. Cloudinary adapter in the Library

By browsing the tree, the *content-hub-adapter-cloudinary* provides an import mechanism for content objects. The screenshot below shows the button for creation.

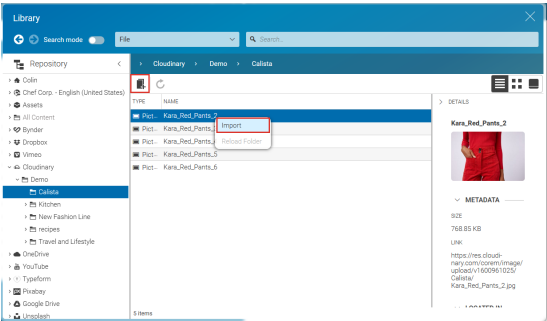


Figure 16.23. Creating content from Cloudinary items



# 16.9 CoreMedia Content Hub Adapter – CoreMedia Repository

The *coremedia-contenthub-adapter* is providing a browsable tree of the current (preferred) site. The image below shows the result after successfully configuring the adapter.

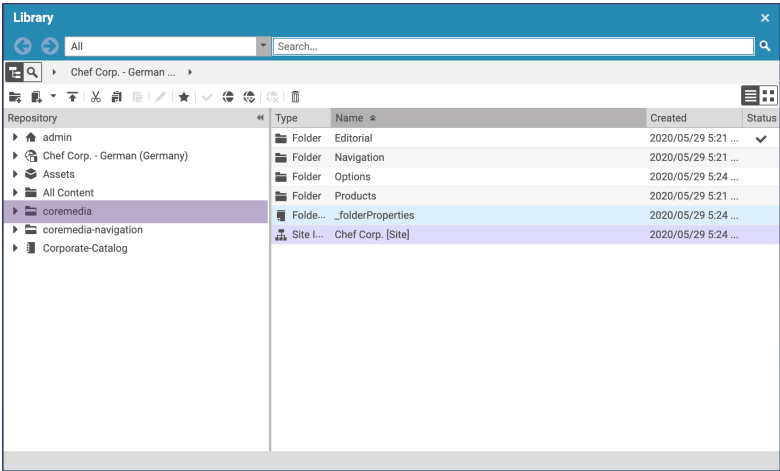


Figure 16.24. Repository adapter in Library

## Basic Configuration

The navigation adapter is coming with some minor configuration options (basically the common).

### Basic Adapter Configuration

This section covers the two possibilities to enable the *content-hub-adapter-filesystem* adapter. Please note that those options are valid for all Content Hub adapters. Before configuring the adapter, refer to the documentation [Content hub configuration](#) for preliminary steps.

### Global Adapter Configuration

To enable the *content-hub-adapter-coremedia* for all sites, create a CMSettings content item at the `contenthub.studio.global-configuration-path` location.

For convenience reasons, it is recommended to name this content item "core-media" (postfix of the extension name).

Site Specific Adapter Configuration

To enable the *content-hub-adapter-coremedia* for a single site, create a CMSets-tings content item at the `contenthub.studio.site-configuration-path` location.

For convenience reasons, it is recommended to name this content item "Filesystem" (postfix of the extension name).

Detailed Adapter Configuration

Basic Structure

The table below shows the initial toplevel entry for all content-hub-connector configurations.

Key	Type	Required
connections	StructList	Yes

Table 16.7. Toplevel entry

After creation of the initial struct list called *connections* the next step is to create the first entry. This can be done in *Studio* with the struct editor by pressing *add Item to ListProperty*. The table below shows the entries which are common for all connectors.

Key	Type	Value	Required
connectionId	String	<YOUR_CHOOSEN_ID>	Yes
factoryId	String	coremedia-navigation	Yes
enabled	Boolean	true or false	Yes
settings	Struct		Yes

Table 16.8. Entries for all connectors

# 16.10 Social Media Hub

The Social Media Hub allows to integrate various social networks into CoreMedia Studio. It provides a separate tab that shows different social network feeds and messages that have been scheduled for publishing.

The Social Media Hub is implemented as a Blueprint extension.

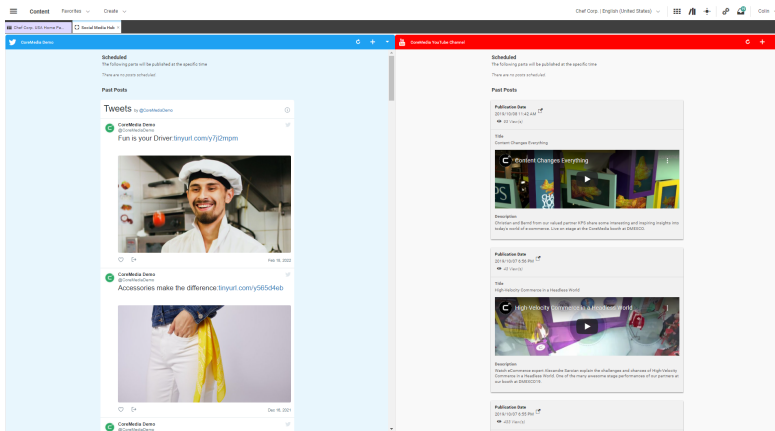


Figure 16.25. CoreMedia Social Media Hub

## Supported Networks

The Social Media Hub comes with a set of predefined adapter and connector implementations. Native adapters are communicating with the social network directly via their (REST) APIs.

The more common use case is that CoreMedia only defines a stub for a social media network and implements a connector that takes care of pushing the composed messages to this tool.

CoreMedia recommends to implement a SocialHubConnector for a social media tool!

The existing integrations of Twitter and YouTube are just example implementations. The philosophy behind the Social Media Hub is to prepare content for social media publication by pushing content items to social media tools which take care of the actual publication.

The following table shows the currently supported features for the various social networks.

Social Network	Publication	Composing
Twitter	✓	✓
Youtube	✓	✓
Instagram	-	✓
Pinterest	-	✓

Table 16.9. Supported Networks

For more details, see here: <https://github.com/CoreMedia/coremedia-social-hub>

## 16.11 Translations with GlobalLink Connect Cloud

This open-source workspace enables CoreMedia CMS to communicate with GlobalLink Connect Cloud (GCC) REST API in order to send contents to be translated, query the translation status and to update contents with the received translation result eventually.

Assuming that you are familiar with the CoreMedia Studio and that you created a new campaign in the English master site that has now to be translated into French, German, and Spanish. This guide shows how this task can be accomplished by means of the GlobalLink Connect Cloud connector.

The connector adds the following functionality to CoreMedia Studio:

- Send content to GlobalLink for translation into one or multiple languages with individual due dates in one or multiple workflows.
- Retrieve content from GlobalLink once the translation is finished.
- Automatically detect cancellations of submissions at GlobalLink and cancel the translation workflow in CoreMedia Studio.
- Configure the connection to GlobalLink per site hierarchy.
- Show additional information like the translation status from GlobalLink in CoreMedia Studio.
- Download XLIFF files and import log files in CoreMedia Studio if an error occurs during import.
- Editors in CoreMedia Studio are notified about completion, cancellation, and import and communication errors of a translation workflow with GlobalLink.

### Send Content to GlobalLink

1. Once finished working on the campaigns content, open the Control Room and click the **[Start a localization workflow]** button in the toolbar of the Localization workflows.

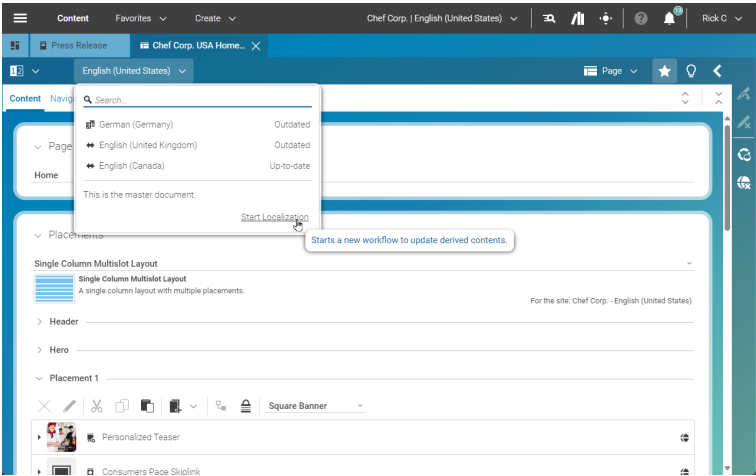


Figure 16.26. Start localization workflow

- 2. In the Start Localization Workflow window ,select the workflow type Translation with GlobalLink, set a self-describing name, a due date, drop the to-be-translated content, and set the target locales.

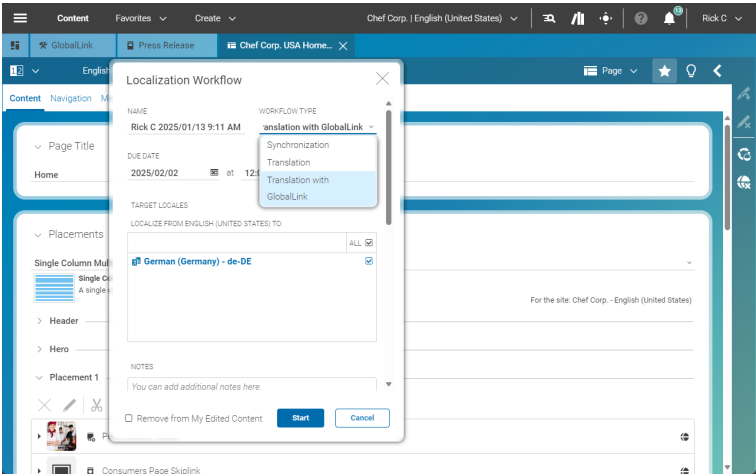


Figure 16.27. Configure workflow

3. After having started the workflow, it is shown in the pending workflow section. Double-click the workflow to open it in the detail view in the Workflow App. Here, you will see the submission state, the ID and other information available.

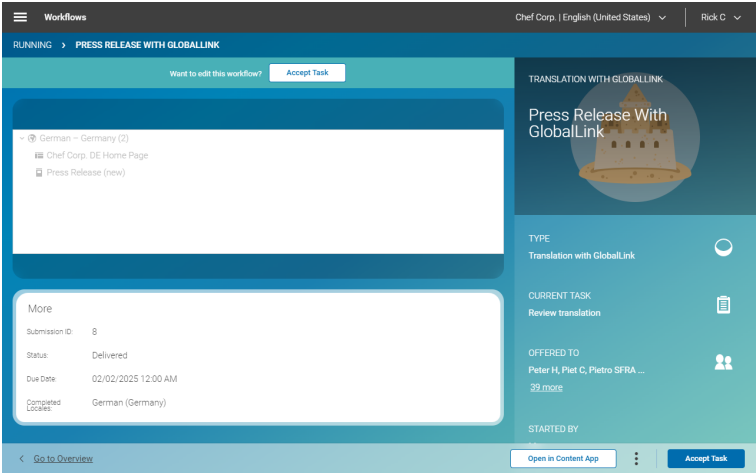


Figure 16.28. Pending workflow is shown

4. In case an error occurs, it is shown in your inbox and you can select to cancel the workflow or you can try to fix the problem and retry.

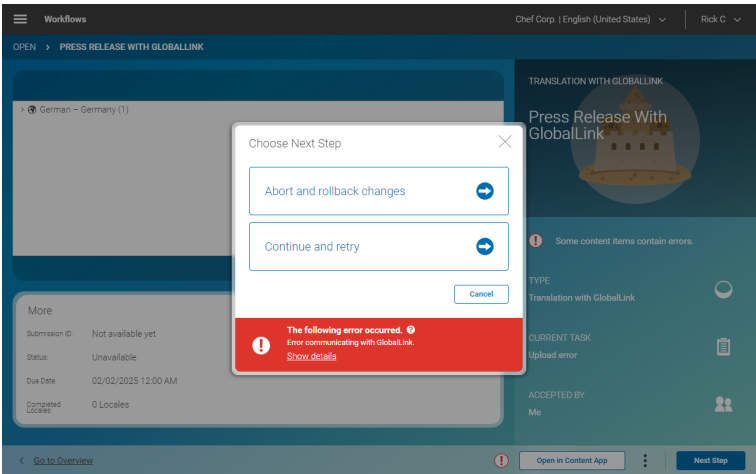


Figure 16.29. Error handling

5. After the translation is finished, you will receive a notification. The workflow is shown in the inbox and once accepting the task, you can review the content and finish it.

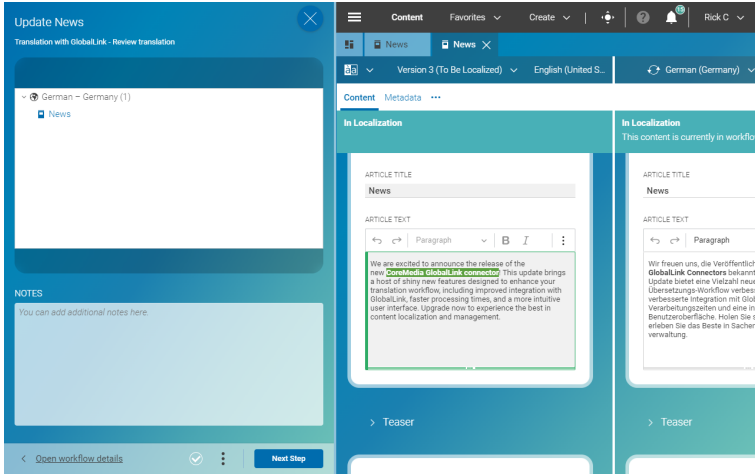


Figure 16.30. Translation is finished

## Configure Connection to GlobalLink Connect Cloud

If the connection is not set up yet, go to **/Settings/Options/Settings/Translation Services/GlobalLink** create a Settings content called GlobalLink and add it to the Linked Settings property of the master site's homepage.



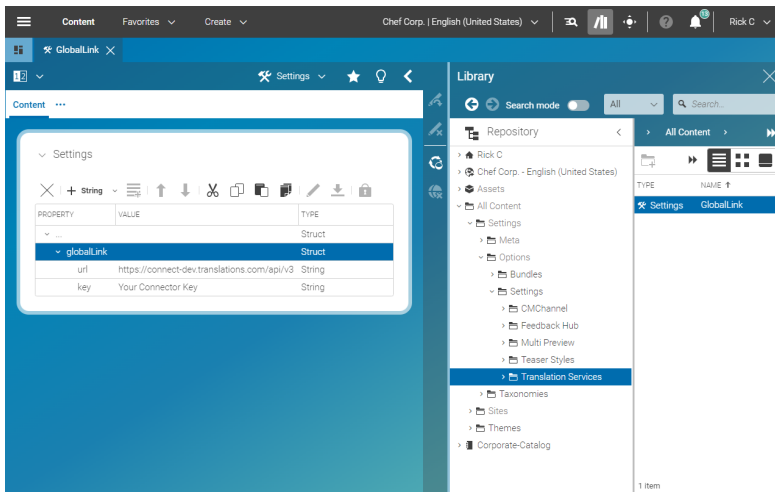


Figure 16.31. Connection to GlobalLink Connect Cloud

## Setup of GlobalLink Connect Cloud

There is some mandatory configuration required in GlobalLink Connect Cloud so that the integration between both systems runs smoothly:

- The connector uses the XLIFF file format to exchange translatable texts with GlobalLink. Make sure that your GlobalLink instance is configured accordingly and request the file format identifier from your contacts at Translations.com.
- The connector automatically detects submissions that have been cancelled in GlobalLink and shows this state to editors in CoreMedia Studio. Canceling individual jobs of a submission is not supported yet and will most likely yield unexpected results. To prevent this from happening, make sure that GlobalLink Cloud Connect only allows the cancellation of submissions.
- Re-opening of submissions is not yet supported by the connector. Ask your contacts at Translations.com to disable this functionality in GlobalLink Connect Cloud to avoid misunderstandings.

For more details, see here: <https://github.com/CoreMedia/coremedia-globallink-connect-integration>

## 16.12 Monetate Integration

The integration of Monetate – a global leader in personalization software for customer facing brands who has rich AI and machine learning capabilities – is the latest addition to CoreMedia's personalisation engine. Testing and segmentation, which are two major aspects for optimizing content, have thus become much simpler and more intuitive.

Two major solutions of the Monetate platform were integrated in CoreMedia Studio: Segments and Experiences.

### Segments

Let's assume, you run an online store for kitchen supplies and you have mainly two categories of customers: professional cooks and home cooks. The former are probably interested in products for the professional kitchen and the latter are more likely to buy something for daily use at home. It therefore is not sufficient to offer the same content to each of them. You will only reach a fraction of them, which considerably can reduce the success of the shop.

This is where segments come in. They help you to offer the right content to the right customers. The CoreMedia personalisation engine enables editors to use segments that were defined in Monetate to tailor the shopping experiences to their customers needs.

In Studio, the editor would be presented with the following view. The professional cook and the home cook are offered products that match their interests, which makes it more likely that they will buy them in your store. Not to forget the customers who do not correspond to any of the specified categories – for example those who visit a site for the first time. For these customers there is the *Baseline*. Here you can place content that appeals to the widest possible audience.

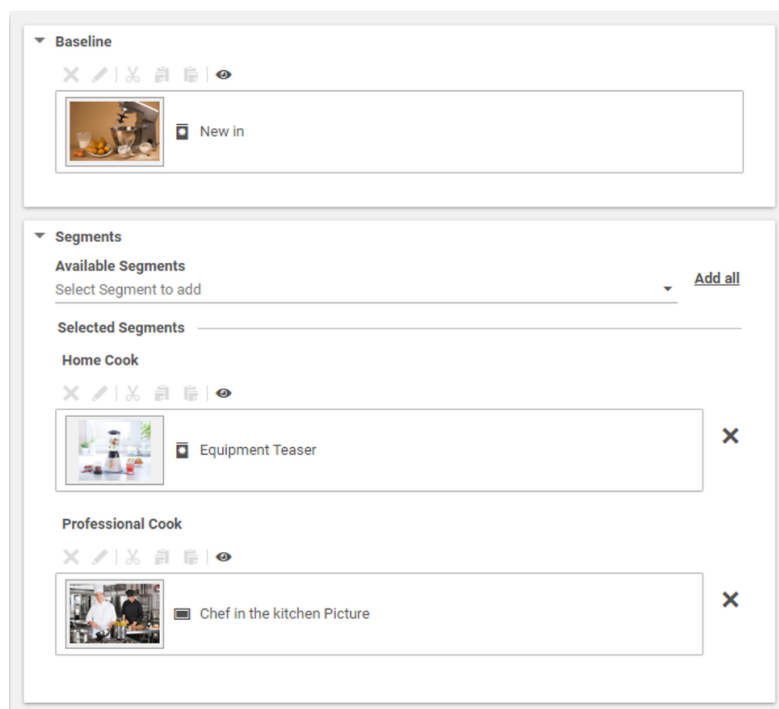


Figure 16.32. Content for different customers

Before the personalized content goes live and is made available to your customers, it is of course important to see what the site would look like from the perspective of the individual segments. This can be done using the new preview function of the CoreMedia personalization engine. Just switch between the various segments and get an impression of the different variants.

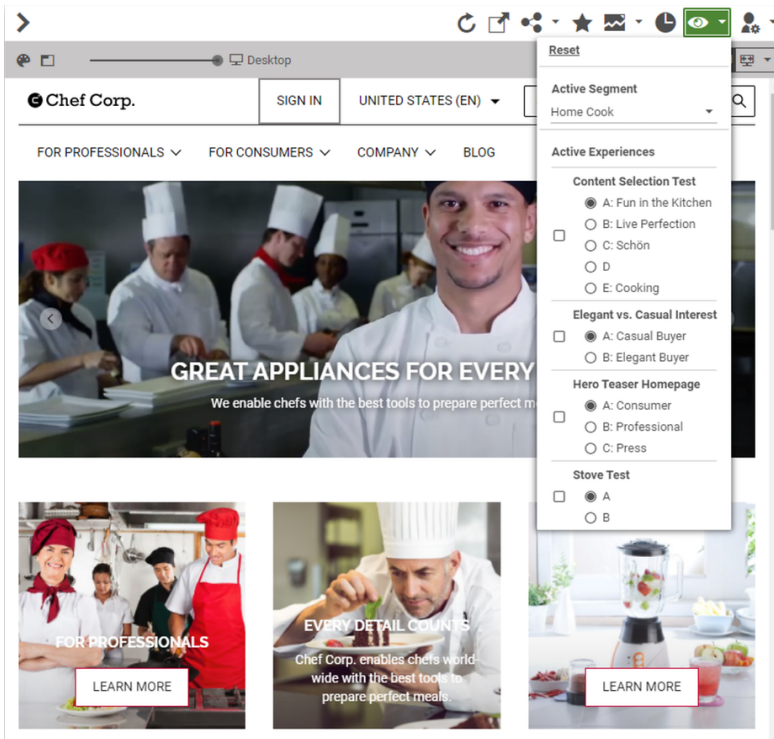


Figure 16.33. A/B testing

Now the content can finally be published. Monetate computes in the background which content is presented to which visitor. Editors do not have to worry about this. They only create the shopping experience for their customers.

## Experiences

There are many different use cases for Monetate experiences, but testing and optimization are probably the most common ones. The concept is simple: First, you create at least two variants of a content item, which differ by various criteria. A randomly selected audience is now offered one of each variant to determine which of them resonates best. This variant can be declared the winner and then be used permanently on the website. Thus the content of your page can constantly be adapted to the needs of the website visitors.

Like segments, experiences are created in Monetate and can then be integrated in Studio by the editors. The editors do not care about how the decisions are

made. They only determine what content is displayed for each variant of an experience. Then they can see how the page will look when a particular variant is selected in the preview.

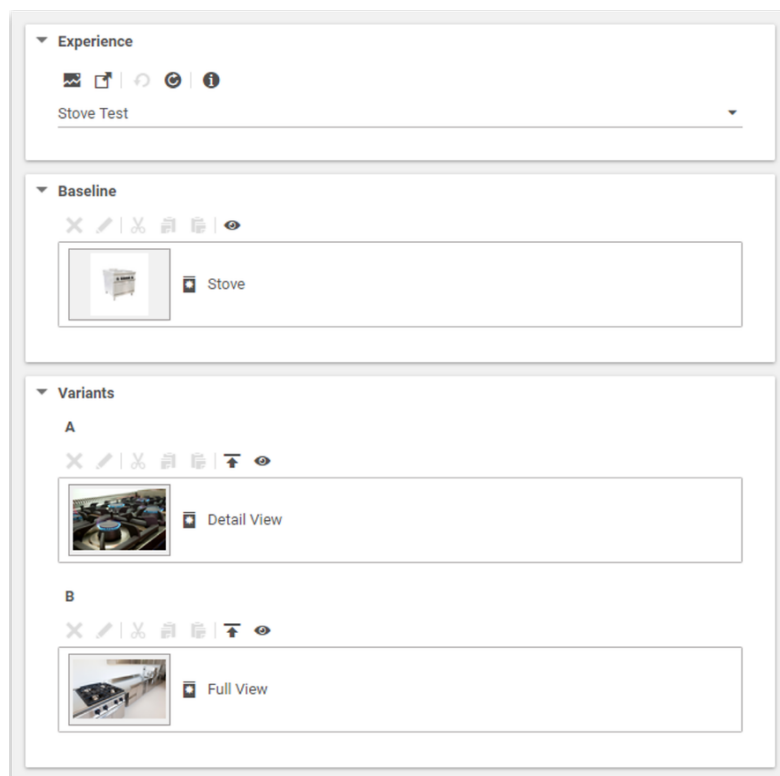


Figure 16.34. Teaser for A/B testing

Once published, the test is active. Monetate not only selects the variants in the background that the respective visitor of the page sees, it also tracks their behavior over a certain period of time. In the experience content item in Studio there is a link that navigates to the tracking results that are available in Monetate. They can be used to determine the variant that resonates best.

## 16.13 Spark – The React Example Application

Spark is a CoreMedia example application based on React, TypeScript and the Headless Server of CoreMedia Content Cloud.

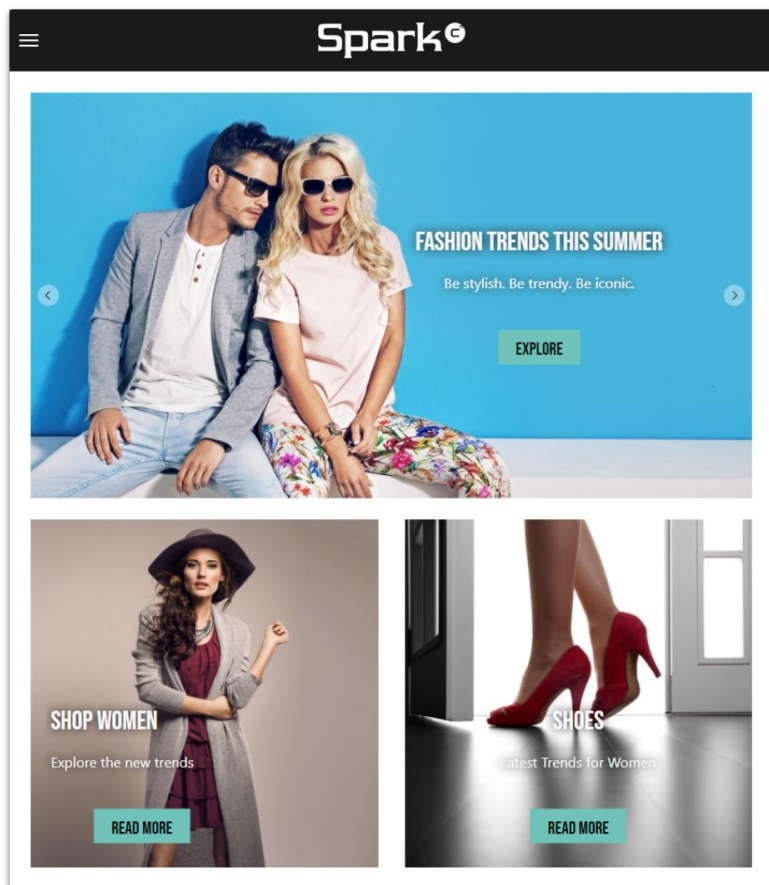


Figure 16.35. Teaser for A/B testing

CoreMedia customers can use Spark as a quick start for building prototype applications. Note that CoreMedia does not recommend the use of these prototype apps in production environments.

This CoreMedia Labs project includes two React applications:

- `app`

Provides guidance for frontend developers that are building websites and other applications based on the CoreMedia Headless server with code examples and documentation on:

- Preview Driven Editing
- Time Travel
- Placement Highlighting
- Content Search
- Fragment Preview and many others

- `app-standalone-fragment`

Demonstrates the ability to load fragments from a CoreMedia Content Cloud and render it on an external website. For example, by injecting resources client side through a tag management solution like the Google Tag Manager.

Although CoreMedia chose React to build Spark with, the principles can easily be transferred to other component libraries and frameworks such as Vue and Angular.

# 16.14 Scheduled Publication Workflow

With the Scheduled Publication Workflow you can simply publish content at a certain time in the future. The following instructions show how this task can be accomplished by means of the publication workflow extensions.

## Start the Workflow

After having completed a new article, you can start publication workflows using the context menu on the content item in the library, as shown here:

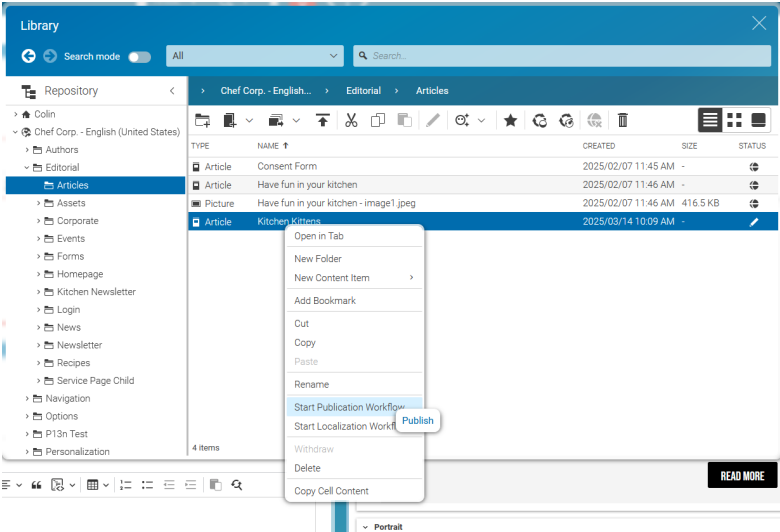


Figure 16.36. Start Publication Workflow

## Schedule the Publication

Choose *Scheduled Publication* from the *Workflow Type* dropdown, and select the *Scheduled Date*. Click on *Start* to start the workflow instance.



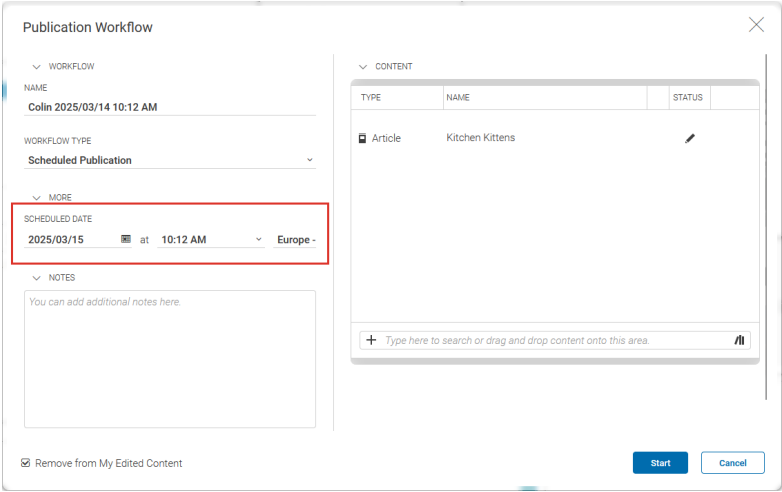


Figure 16.37. Select scheduled date

## Monitor the Scheduled Publication

While the scheduled publication is still pending, it is listed under the pending workflows. Notice that as long as it is pending, the content item is locked so that it cannot be edited.

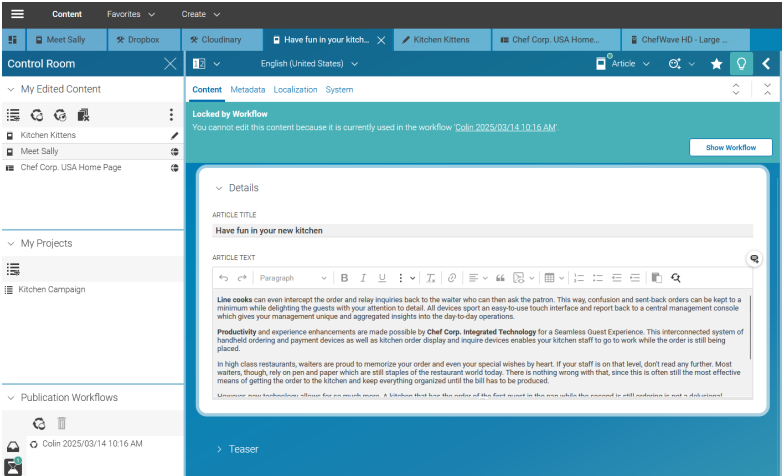


Figure 16.38. Pending scheduled workflow

By double-clicking on the workflow instance it opens in the Workflow App. Here you get the details of the workflow. Most importantly, you can see the scheduled date of the publication.

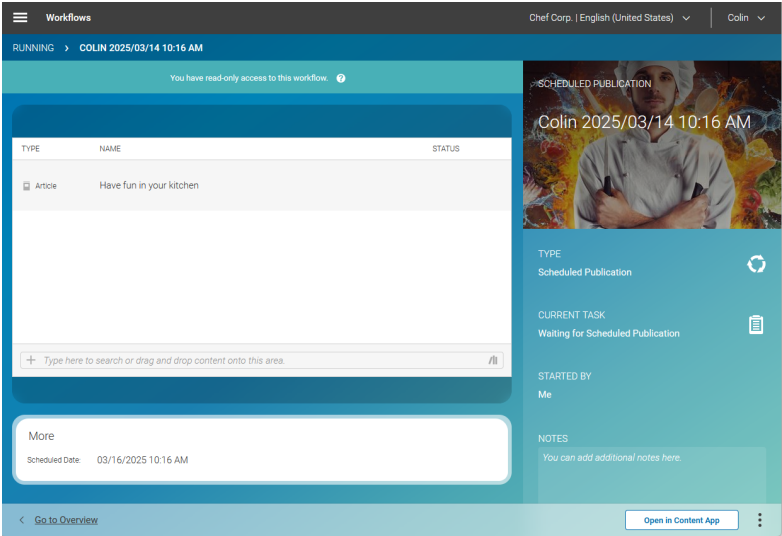


Figure 16.39. Checking the workflow

## Completion of the Scheduled Publication

When the scheduled date of the publication expires, the content item will be automatically published. The workflow instance is no longer pending and is now listed under the finished workflows:

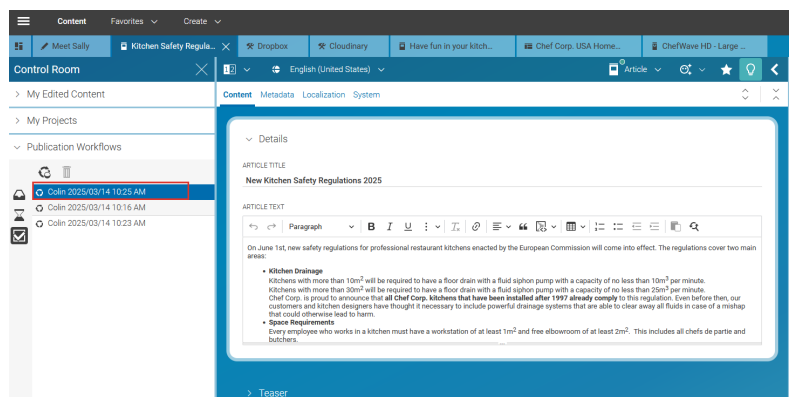


Figure 16.40. Completed workflow

Notice that the content item is no longer locked and can be edited.

Well done. The content item is now published!

# 16.15 Three-Step Publication Workflow

When you have created a new article with CoreMedia Studio which should be sent to another editor for review who will then forward it to publication to a third editor, you can use the three-step publication workflow from the three-step publication workflow extension.

## Start the Workflow

After having completed a new article you can start publication workflows using the context menu on the content item as shown here:

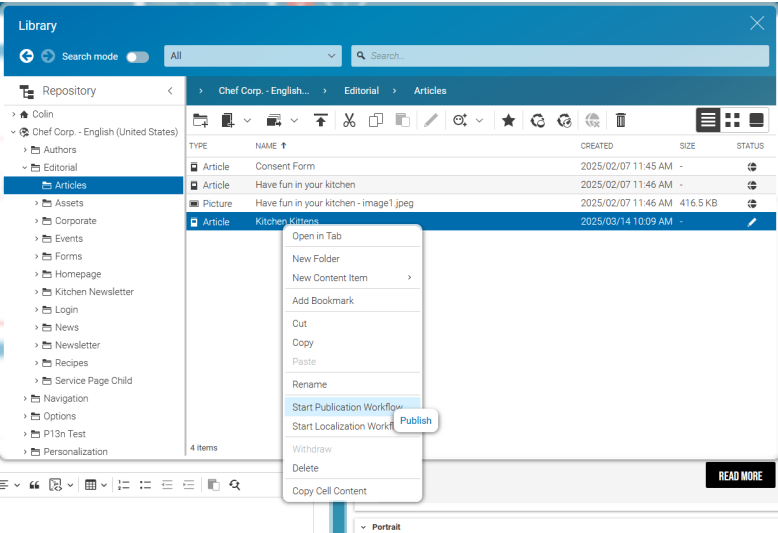


Figure 16.41. Start Publication Workflow

## Start the Three-Step Publication Workflow

Choose *Reviewed and Confirmed Publication* from the *Workflow Type* list.

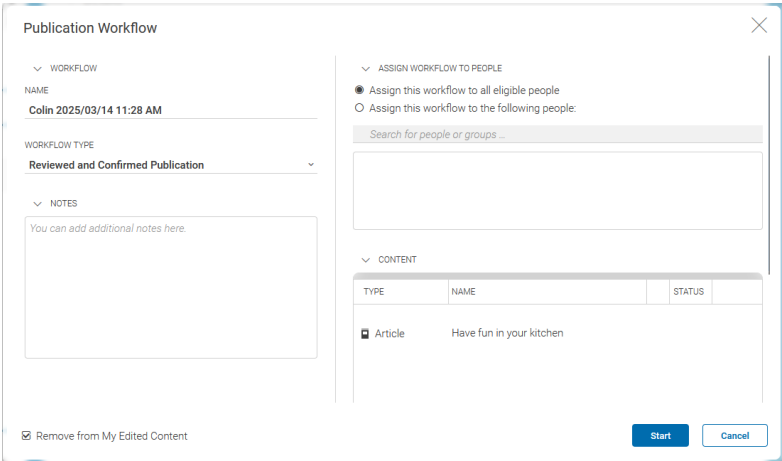


Figure 16.42. Start Three-Step Publication Workflow

## Assign Workflow To People

Assign the task of approving the content item to all other editors by simply selecting the option *All Eligible People* or by specifying one or more editors:

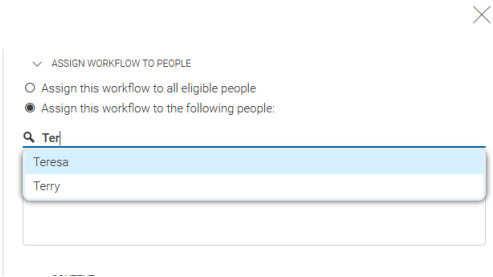


Figure 16.43. Assign workflow for approving

Click **[Start]** to start the workflow instance.

## Monitor the Three-Step Publication Workflow

While the three-step publication workflow is yet pending, it is listed under the pending workflows. Notice that as long as it is pending, the content item is locked so that it cannot be edited.

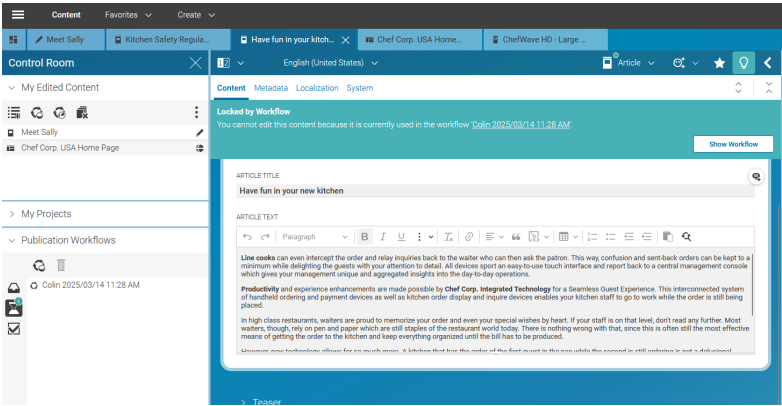


Figure 16.44. Pending workflow

By double-clicking on the workflow instance it opens in the Workflow App. Here you get the details of the workflow. Most important, you can see the editors the workflow instance is assigned to.

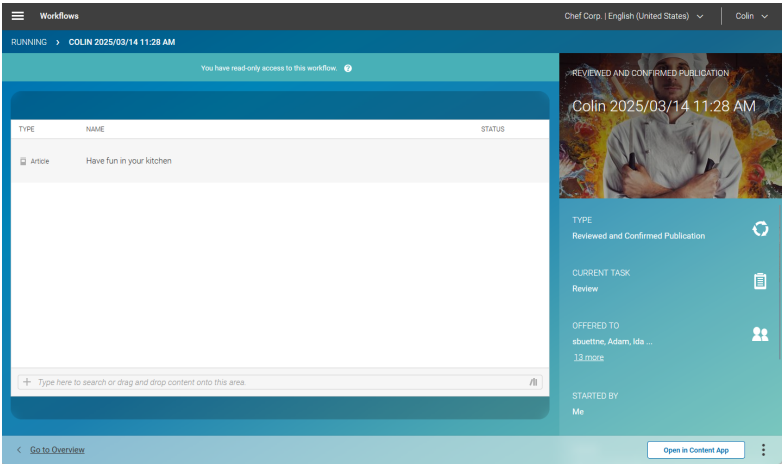


Figure 16.45. Details of workflow after start

## Approve Step of the Three-Step Publication Workflow

For editors who the workflow instance is assigned to, it is now listed under the inbox of workflows. Note that the instance's title contains *Review* as the task to be performed.

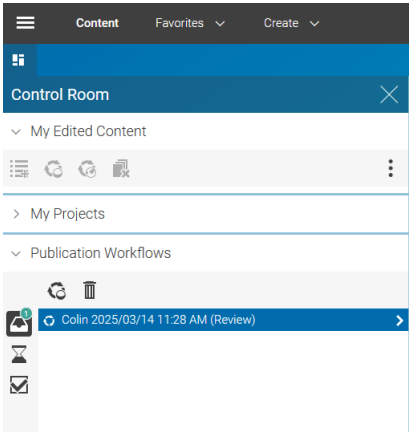


Figure 16.46. Workflow in inbox for review

By double-clicking on the workflow instance it opens in the Workflow App. You see the workflow details, that is, who has started the workflow and which content item has to be reviewed. Accept the task when you are ready to review and approve the content item.

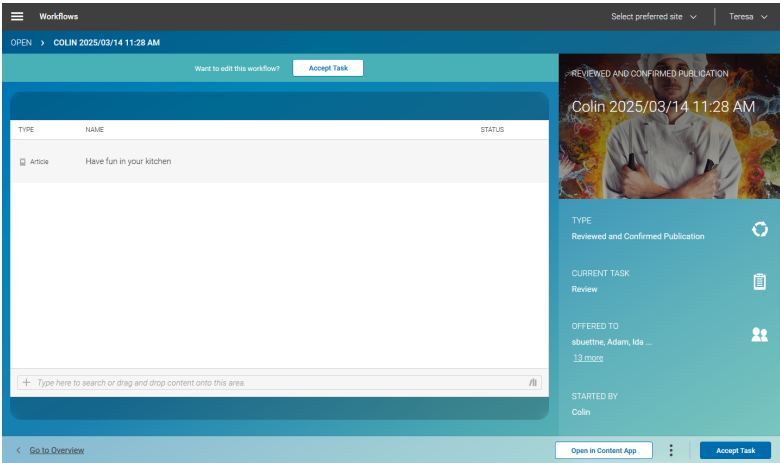


Figure 16.47. Details of the workflow before review

You can now review the content item. When everything is fine select **[Confirm]** as the next workflow step and assign the task to yet another editor like Adam in this example.

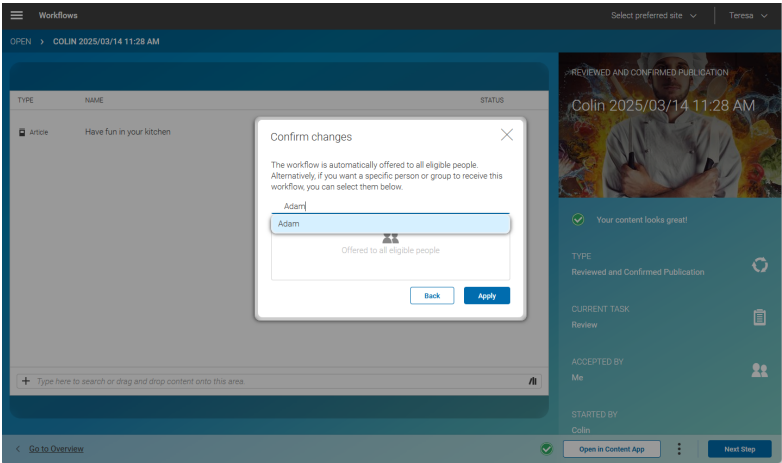


Figure 16.48. Assign workflow to people

Click **[Apply]**; the workflow instance moves from the inbox to the pending list.

Publish Step of the Three-Step Publication Workflow

Editors who the workflow instance is assigned to – Adam in this example – see the workflow now in the inbox of workflows. Note that the instance’s title contains *Confirm* as the task to be performed.

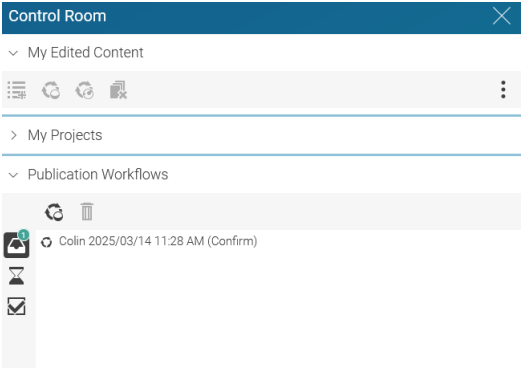


Figure 16.49. Workflow in the inbox

By double-clicking on the workflow instance it opens in the Workflow App. You see the workflow details, that is, who has started the workflow and which content



item is to be published. Accept the task when you are ready to review and publish the content item.

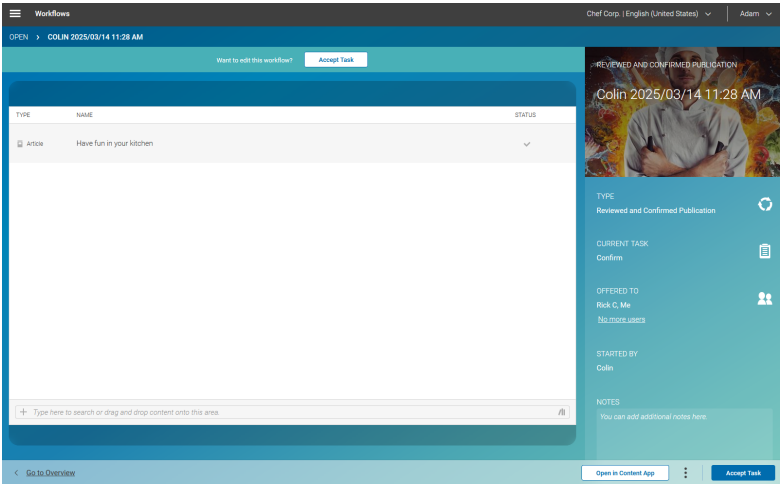


Figure 16.50. Details of the workflow before publication

When everything is fine, and the content item can be published, click **[Next Step]**. Click *Publish* and confirm with **[Yes, continue]** to publish the content item.

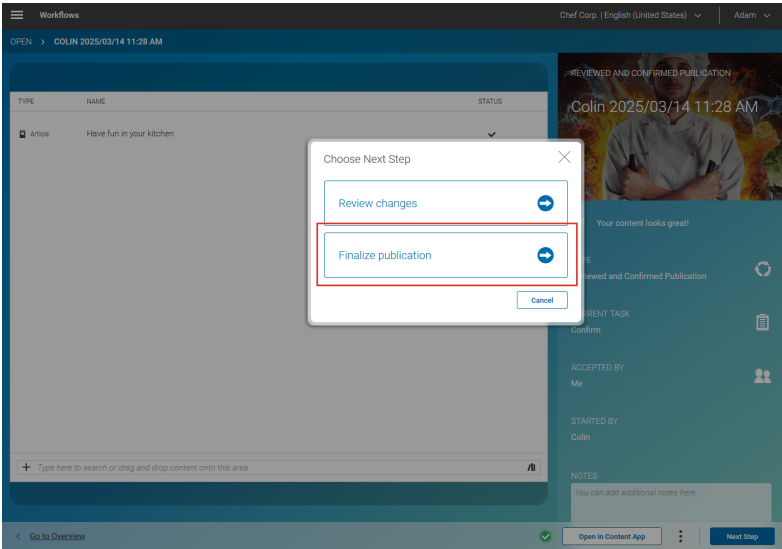


Figure 16.51. The publish step

Now, the content item is published.

# 17. CoreMedia Engagement Cloud

CoreMedia Engagement Cloud is part of CoreMedia Experience Platform along with Content Cloud. Both clouds work standalone or integrated. When working integrated, they improve your experience and augment the available functionality giving you a 360° view of your customers.

The following Studios are part of Engagement Cloud:

## Engagment Studio

Use Engagement Studio to manage campaigns, segments, and profiles.

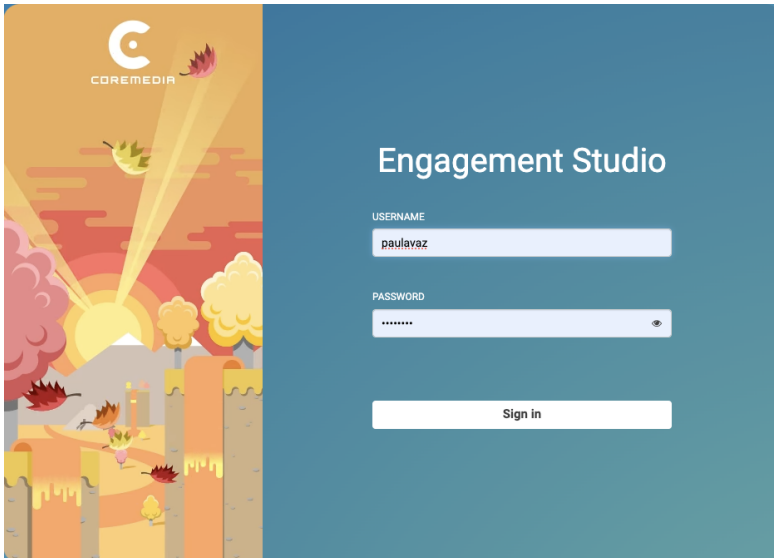
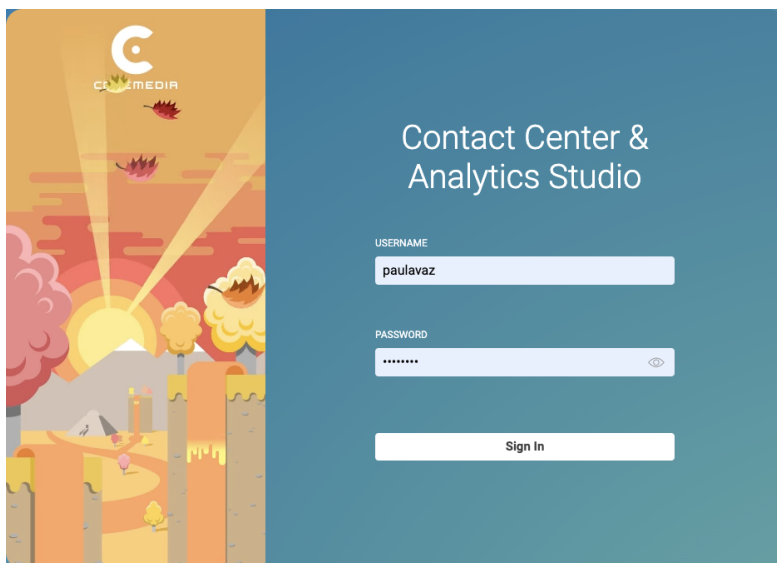


Figure 17.1. Engagement Studio

## Contact Center and Analytics Studio

Use Contact Center and Analytics Studio to manage users, teams, and analytics.



*Figure 17.2. Contact Center and Analytics Studio*

## Chatbot and Automation Studio

Use Chatbot and Automation Studio to manage chatbots, automations and AI integrations.

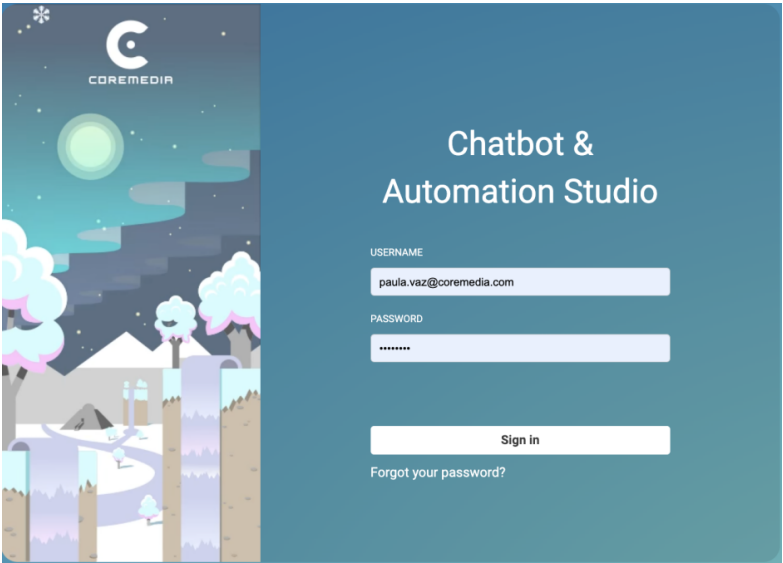


Figure 17.3. Contact Center and Analytics Studio

# 17.1 CoreMedia Engagement Cloud Integrations

As we embrace the digital age, we find ourselves at the forefront of a transformative era in the digital landscape. Across the globe, businesses are on a mission to redefine the customer experience, transcending the mere delivery of products and services. The goal of this evolution is to establishing genuine connections with individuals at every touch point of their journey.

At CoreMedia, we are unwavering in our commitment to fostering these meaningful connections. Our integration solutions play a pivotal role in this endeavor. We understand that the contemporary business landscape demands holistic, multifaceted experiences that span the realms of sales, service, marketing, and beyond. CoreMedia stands as the vanguard of customer engagement platforms, empowering organizations to seamlessly integrate and unite these customer experiences across every facet of their operations.

In today's interconnected and rapidly evolving digital environment, integrating with external providers has evolved into a strategic necessity for businesses spanning various industries. Whether you're a tech startup, a seasoned enterprise, or a dedicated service provider, the integration of your platform with external providers offers a diverse array of advantages. These integrations become a driving force for innovation, an avenue to enhance functionality, and a catalyst for an enriched user experience.

This document serves as a guide to elucidate the primary reasons and benefits of such integrations, emphasizing why they should constitute an essential element of your overarching business strategy. At CoreMedia, we combine our commitment to meaningful connections with the power of external provider integrations, enabling you to chart a course towards a more engaging and prosperous digital future.

## 17.1.1 What is an Integration?

An integration, in the context of technology and software, refers to the process of connecting different systems, applications, or components to work together and share data or functionality. Integrations enable these disparate systems to communicate with each other and exchange information in a seamless and automated manner.

Key characteristics of integrations include:

- **Data Exchange:** Integrations often involve the exchange of data between systems. This can include sending, receiving, or synchronizing data in real-time or through batch processes.
- **Functionality Sharing:** Integrations can also involve sharing specific functions or services between systems. For example, an e-commerce website may integrate with a payment gateway to process payments.
- **Automation:** Integrations automate processes that would otherwise require manual data entry or redundant tasks. This can improve efficiency and reduce the risk of errors.
- **Interoperability:** Integrations allow different technologies or applications to work together seamlessly. This is especially important in complex software ecosystems.
- **Customization:** Many integrations can be customized to meet specific business needs. This may involve configuring data mappings, triggers, and actions.

Examples of integrations include connecting a customer relationship management (CRM) system with an email marketing platform, integrating an e-commerce website with a shipping and logistics system, or linking an enterprise resource planning (ERP) system with an inventory management system.

### 17.1.2 Why Integrating with External Entities or Providers?

Integration with external entities or providers for data sharing and retrieval is a strategic move that can enhance your platform's functionality, improve user experiences, and drive cost efficiencies, ultimately leading to a more competitive and successful offering.

- **Data Access:** External providers often have valuable data or services that can enhance your platform. Integration allows you to access and utilize this data effectively.
- **Specialized Services:** External providers may specialize in specific areas, such as payment processing, geo-location services, or social media data. Integrating with them gives you access to these specialized services without developing them in-house.
- **Data Enrichment:** Data from external sources can enrich your platform, providing more comprehensive and accurate information to users.
- **Improved Decision-Making:** Access to external data can lead to better-informed decisions and more personalized user experiences.

- **Cost Efficiency:** Integrating with external providers can be more cost-effective than developing and maintaining similar services internally.
- **Time Savings:** Integration reduces development time and accelerates the deployment of new features and functionalities.
- **Scalability:** As your user base grows, external providers can often scale their services to meet increased demand, ensuring your platform remains responsive.
- **Focus on Core Competencies:** Integration allows your team to concentrate on core features unique to your platform, rather than investing time in non-core services.
- **Stay Current:** External providers continuously update and improve their services, ensuring your platform remains up to date and competitive.
- **Enhanced User Experience:** Integrating external data can lead to a richer and more satisfying user experience, increasing user engagement and satisfaction

### 17.1.3 CoreMedia Integrations

In line with the principles of integrability and composability (where composable systems are built in a way that components can be easily connected, rearranged, or replaced), CoreMedia is committed to enhancing efficiency, delivering superior experiences, and enabling effortless customization and adaptation of our solutions to meet the unique needs of our customers.

CoreMedia has the following methods to integrate solutions (the choice depends on the specific project requirements and the technology involved):

- **REST API (Representational State Transfer Application Programming Interface):** CoreMedia REST API provides programmatic access to some of the CoreMedia functionalities and services.
- **Javascript API:** Enabling client-side interaction with CoreMedia platform.
- **Webhooks:** Webhooks are HTTP-based triggers that allow one application to automatically send information to another when specific events occur. They are useful for notifying external systems about real-time actions.
- **CoreMedia SFTP Service:** Exchange of files – contact lists, reports – in a secure environment.
- **CoreMedia Integration app:** Pre-packaged integration connectors built in CoreMedia platform and available to be setup and enabled within a App.
- **CoreMedia Export App:** App responsible to export CoreMedia data



- **CoreMedia Custom App:** Custom-built applications within CoreMedia's backoffice for direct integration with internal systems, enhancing functionality tailored to specific client needs.
- **CoreMedia DataSources:** Connects visitor profiles to client systems, enabling real-time data enrichment from various sources.

The choice of integration approach depends on the project's complexity, the systems involved, the integration objectives and where the touchpoint need to be triggered. Often, a combination of these methods is used to create a comprehensive integration solution.

### 17.1.3.1 CoreMedia Events

CoreMedia's event-oriented architecture is a powerful and flexible solution designed to enhance data integration and actions with external providers. It prioritizes real-time responsiveness, scalability, and adaptability, making it an ideal choice for organizations seeking to streamline operations and provide top-notch customer experiences.

This architecture facilitates real-time data exchange, ensuring that interactions and updates happen immediately for up-to-the-minute information and rapid responses. It is highly scalable, making it suitable for growing businesses with dynamic customer interactions. Additionally, its flexibility allows easy adaptation to evolving business requirements and integration needs without major architectural changes.

Asynchronous processing enhances system performance and prevents bottlenecks, while fault tolerance ensures continuous data flow. Loose coupling enables system components to operate independently, reducing the risk of disruptions caused by changes or issues in one part of the system.

CoreMedia's event-driven architecture excels at integrating data and actions with various providers, both internal and external, offering versatility and adaptability to diverse data sources. Automated triggers streamline processes, boost efficiency, reduce manual intervention, and enhance the customer experience.

Real-time data integration ensures data accuracy and timeliness, crucial for informed decision-making and top-notch customer service. Additionally, monitoring and analytics tools included in the platform allow organizations to track event flow and gain insights into system performance.

In summary, CoreMedia's event-oriented architecture is a robust solution designed to elevate data integration and action with external providers. It offers real-time responsiveness, scalability, and adaptability, enhancing both customer experiences and operational efficiency.

## 17.1.4 CoreMedia REST API

The CoreMedia API is one of several web interfaces integration touch points that enable you to access your CoreMedia data without relying solely on the user interface. With API access, you have the flexibility to perform various operations and seamlessly integrate CoreMedia into your applications as needed.

By using the CoreMedia API, you can effortlessly create, manage, and search data within the CoreMedia platform by sending HTTP requests to specific end-points using all the security controls we have in place in the platform. These endpoints are responsible for providing access to various types of information known as resources. Resources in the CoreMedia API allowing you to interact with a wide range of data and functionalities.

The CoreMedia API adopts RESTful architecture, which results in a simple and consistent interface. One of the primary advantages of the CoreMedia API is its ease of use, requiring minimal tooling to access your data.

## 17.1.5 Javascript API

The CoreMedia JavaScript API, accessible upon the installation of the CoreMedia tag, provides a gateway to a variety of system functionalities, enhancing site interactivity and personalization based on visitor behavior. Key functionalities include:

- **Initiating Calls:** Empower your site to start phone calls directly, fostering immediate and seamless communication.
- **Triggering CoreMedia Automations:** Activate CoreMedia automations that facilitate interaction with external systems, thereby bridging the gap between your digital presence and the external world.
- **Segment Management:** Add or remove visitors from specific segments, allowing for more targeted audience engagement and refined marketing strategies.
- **Enriching Visitor Profiles:** Append additional information to a visitor's profile, enriching their data footprint and enabling more personalized interactions.
- **Activating Campaigns (Interfaces) for Visitors:** Crucially, the API allows for the activation of campaigns or interfaces for visitors at specific moments. For instance, if you wish to present a campaign to a visitor at a particular point in a form, this can be programmed via JavaScript to display precisely when needed.

- **Event Tracking for E-commerce Insights:** Send events to track e-commerce activities, providing valuable insights into customer interactions. This feature is pivotal for understanding on-site behavior, like items added to a cart.
- **Data-Driven Reactions:** The API empowers the system to react based on this data. For example, it can inform an operator about the contents of a customer's shopping cart or trigger specific campaigns if the cart value exceeds a certain threshold.

To use this api, it is important to be familiar with CoreMedia events, that should be used to inform CoreMedia platform about data, behavior, ... Events must be set up in advance within the CoreMedia backoffice.

Example code for appending information to a visitor profile is as follows:

```
html script type="text/javascript"> {function(){ <var event_info = new
Object(); event_info.name =
"name"; event_info.email = "email"; event_info.context = "context";
event_info.segment = "segment";
bySideWebcare_event("SetVisitorInfo", event_info); }}();
</script>
```

*Example 17.1. Appending information*

**NOTE:**

The interface provides codes like the example above, ready to be executed on the site at the appropriate time when the information becomes available.

## 17.1.6 CoreMedia SFTP Service

CoreMedia's Secure File Transfer Protocol (SFTP) service offers a robust and secure solution for data transfer, operating independently of CoreMedia's core to ensure data integrity and security during transmission.

This service has the following key Features:

- **Secure Data Transfer:** Ideal for transferring large data volumes like contact lists and reports, ensuring protection against unauthorized access.
- **Isolation from CoreMedia Core:** The service operates isolated from CoreMedia's core, adding an extra layer of security and minimizing risk to core operations.
- **Immediate File Processing:** Files received via SFTP are processed instantly, crucial for operations requiring real-time data updates.
- **Automatic Deletion Post-Processing:** Ensures files are automatically deleted from the server after processing, securing sensitive data.

A good example of how this service can be used is a customer that wants to automate daily email marketing dispatches. CoreMedia sets up a dedicated SFTP space for this purpose. When a file is uploaded to this space, it is immediately imported into an email campaign system, and emails are dispatched instantly. This process is especially efficient for bulk data operations, ensuring quick and effective email marketing campaigns.

### 17.1.6.1 Technical Details

- **File Types and Limitations:** Supports a wide range of file formats, with size and file type limitations provided during initial setup.
- **Setup and Usage:** Straightforward setup process, including establishing secure connections and file transfer parameters.
- **Security Protocols:** Utilizes advanced protocols to protect data during transfer, maintaining compliance with data security standards.

### 17.1.6.2 Security and Compliance

- **Data Protection:** Top priority is given to data security, with stringent measures protecting against unauthorized access or interception.
- **Compliance with Security Standards:** Adheres to major data security standards, ensuring legal and compliance requirements are met.

CoreMedia's SFTP service is an essential tool for organizations requiring a reliable, secure data transfer solution, particularly beneficial for large-scale operations like email marketing.

## 17.1.7 CoreMedia Integration App

In order to maximize the customer experience, it is necessary for all systems to communicate effectively, unifying information and eliminating silos while optimizing operations. Therefore, we centralize and provide proper visibility to our customers of the integrations that CoreMedia are already conducting.

CoreMedia provides an app to its customers that allows connectivity with the following providers using pre-built connectors embedded on CoreMedia platform:

- Salesforce
- Google Analytics
- Google Ads

- Facebook Offline Conversions
- Campaign Manager 360

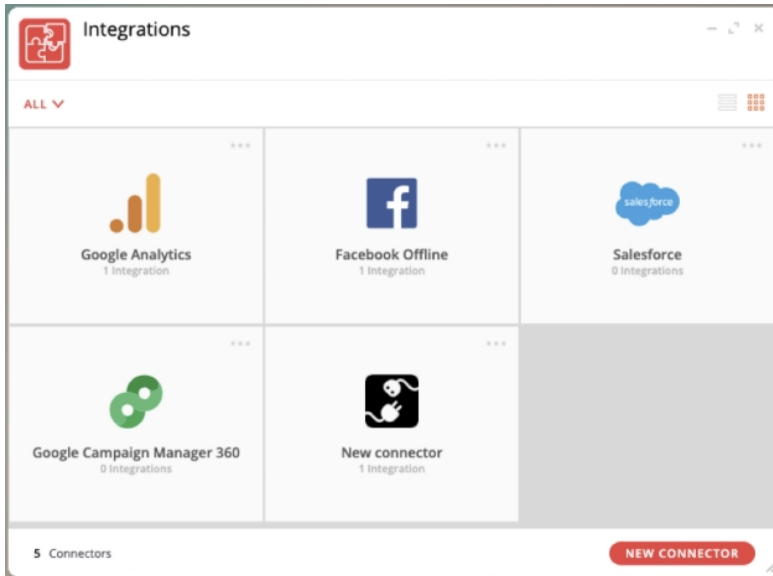


Figure 17.4. CoreMedia Integration App

## 17.1.7.1 Salesforce

Salesforce API Connector is a software component or service that facilitates the integration of Salesforce, a popular Customer Relationship Management (CRM) platform, with other applications, systems, or services.

Salesforce provides various APIs that allow you to send and receive different types of information to and from the platform. The type of information you should send using Salesforce APIs depends on your specific business needs and use cases. Here are some common types of information that can be sent using Salesforce APIs:

- **Lead and Contact Data:** You can use APIs to create, update, and synchronize lead and contact data. This includes information such as names, email addresses, phone numbers, and job titles.
- **Account and Opportunity Information:** Salesforce APIs allow you to manage account and opportunity data, which is essential for managing your sales pipeline and customer relationships.

- **Custom Objects:** You can send and retrieve data related to custom objects that you've created in Salesforce to store specific information unique to your organization.
- **Event and Activity Data:** Salesforce APIs enable you to log and manage events, meetings, and activities associated with leads, contacts, and accounts.
- **Campaign and Marketing Data:** You can use APIs to manage marketing campaigns, track the effectiveness of marketing efforts, and capture data on campaign performance.
- **Case and Service Data:** Salesforce APIs can be used to create, update, and manage cases and service requests, helping your customer support teams.
- **Report and Dashboard Data:** APIs allow you to access data from Salesforce reports and dashboards, which can be used for analytics and reporting in external systems.
- **Custom Application Data:** If you've developed custom applications in Salesforce, you can send and receive data related to these applications through APIs.
- **Authentication and User Data:** APIs are used for authentication and user management, allowing you to manage user accounts and access control.
- **Integration with External Systems:** Salesforce APIs are often used to integrate Salesforce with other external systems, such as marketing automation platforms, e-commerce solutions, accounting software, and more.

To integrate with this api, you need to configure a new connection on CoreMedia integration app, indicate the authentication and mapping the respective actions that should be synchronized.

### 17.1.7.2 Google Analytics

The main goal of integrating data with Google Analytics via API is to expand data analysis and visualization capabilities. The Google Analytics APIs allow users to access and manipulate Analytics data programmatically, which opens up a wide range of possibilities for data analysis.

Here are some specific examples of how Google Analytics APIs can be used to integrate data:

- **Website Performance Analysis:** APIs can gather data of website traffic, user behavior, and page performance captured by other provider and compile the information to understand how users are interacting with your website, identify popular pages, track conversion rates, and optimize user journeys.

- **E-commerce Analysis:** APIs can collect data related to product sales, revenue, and customer behavior to allow an e-commerce performance analyze, track product popularity, monitor cart abandonment rates, and optimize the sales funnel.
- **Marketing data integration:** APIs can be used to integrate marketing data, such as loves, interactions in webpage or social media campaign data, with Google Analytics data. This can help businesses evaluate the impact of their marketing campaigns.
- **Sales data integration:** APIs can be used to integrate sales data, such as lead and conversion data, with Google Analytics data. This can help businesses better understand their customers' behavior.
- **Customer service data integration:** APIs can be used to integrate customer service data, such as ticket and complaint data, with Google Analytics data. This can help businesses improve the quality of their customer service.

The goals of integrating data using the Google Analytics API are typically tied to improving website or app performance, enhancing user experiences, making data-driven decisions, and achieving specific business objectives. The specific information and goals you pursue will depend on your organization's priorities and strategies.

### 17.1.7.3 Google ADS

The Google Ads API is an application programming interface (API) provided by Google that allows developers to interact with Google Ads (formerly known as Google AdWords) advertising campaigns. It provides a way to automate various aspects of advertising campaign management and retrieve valuable data for analysis and reporting. Here are some key features and capabilities of the Google Ads API:

- **Campaign Management:** The API allows you to create, update, and manage advertising campaigns, ad groups, and ads. You can set campaign budgets, bidding strategies, and targeting criteria.
- **Keyword and Ad Management:** You can work with keywords, ad designers, and ad extensions, enabling you to optimize ad copy, headlines, and descriptions.
- **Performance Data:** Retrieve data on ad performance, including metrics like impressions, clicks, click-through rates (CTR), conversions, cost, and more. This data is valuable for assessing campaign effectiveness.
- **Audience Targeting:** Access and manage audience targeting options, including demographics, interests, and behaviors.

- **Conversion Tracking:** Set up and track conversions to measure specific actions taken by users, such as form submissions, purchases, or app installations.
- **Quality Score:** Access quality score data for keywords and ads, helping you gauge the relevance and quality of your campaigns.
- **Competitor Insights:** Gain insights into competitors' ad campaigns, keywords, and bidding strategies, which can inform your own advertising strategy.
- **Custom Reporting:** Create custom reports and dashboards to monitor key performance indicators (KPIs) and campaign goals specific to your business objectives.
- **Ad Extensions:** Manage ad extensions, such as site link extensions, call out extensions, and structured snippet extensions.
- **Geographic Targeting:** Control geographic targeting settings to determine where your ads are displayed.
- **Billing and Budgeting:** Access information related to ad spend, budget allocation, and billing details.

The specific information you send and receive using the Google Ads API will depend on your advertising objectives and the metrics and insights you need to optimize your campaigns. By leveraging this data, you can make informed decisions, adjust your advertising strategies, and improve the performance of your Google Ads campaigns.

At this moment, CoreMedia already use this API to send conversions tracking, to send performance campaign data and audience tracking. To do that, you need to configure a new connection on CoreMedia Integration App, indicate the authentication and mapping the respective actions that should be synchronized.

### 17.1.7.4 Facebook Offline

This integration was replaced by Facebook conversion API because Facebook decommissioned the old API.

"Starting with Graph API v17.0, the Offline Conversions API will no longer support offline events. Graph API v16.0 is the last version that supports offline events. We anticipate that the Offline Conversions API will be discontinued in the third quarter of 2024.

In February 2023, we announced that the Conversions API now fully supports offline events. We recommend that advertisers use the Conversions API for new integrations. We also recommend that advertisers with existing Offline Conversions API integrations convert their integration into a Conversions API integration before the third quarter of 2024 and not update their Offline Conversions API until they have successfully done so. Learn more about the Conversions API."



IN "Facebook Webpage"

## 17.1.7.5 Facebook Conversion

The Facebook Conversion API (Conversions API or CAPI) is a tool provided by Facebook that allows businesses and advertisers to send customer interaction data directly to Facebook's servers from their own servers. This API is designed to complement and enhance the tracking capabilities provided by browser-based methods, such as the Facebook Pixel.

Key features and aspects of the Facebook Conversion API include:

- **Server-Side Tracking:** Unlike client-side tracking methods that rely on browsers and cookies, the Conversions API operates on the server side. This means that data about customer events, such as purchases or form submissions, is sent directly from the business's server to Facebook.
- **Enhanced Privacy Compliance:** As privacy regulations evolve, there are increasing limitations on browser-based tracking methods. The Conversions API can help businesses comply with privacy regulations by providing an alternative method of sending customer event data without relying solely on cookies.
- **Cross-Device Tracking:** The Conversions API helps improve cross-device tracking accuracy. By sending data directly from the server, businesses can gain a more comprehensive view of customer interactions that occur across multiple devices.
- **Offline Events Tracking:** Businesses can use the Conversions API to track offline events, such as purchases made in physical stores, phone orders, or other interactions that are not solely online. This feature allows advertisers to measure the impact of their online campaigns on offline conversions.
- **Custom Event Tracking:** The API supports the tracking of custom events and actions that are specific to a business's objectives. This flexibility allows businesses to define and track events that matter most to them.
- **Reduction of Data Loss:** Server-side tracking can help reduce the likelihood of data loss associated with factors such as ad blockers or browser settings that may affect client-side tracking.
- **Enhanced Data Accuracy:** Direct server-to-server communication can provide more accurate and reliable data compared to relying solely on browser-based tracking methods.

It's important to refer to Facebook's official documentation for the Conversions API to get detailed information on implementation, supported events, and any updates or changes to the API.

At this moment, CoreMedia already use this API to send offline conversions and track special events. To accomplish this, you should set up a new connection in the CoreMedia Integration App, specifying the authentication, and mapping the corresponding actions for synchronization.

### 17.1.8 CoreMedia Export App

CoreMedia Exports App allows our clients to obtain information about their business easily and intuitively, either sporadically or periodically. At this moment customer only can export data from byside analytics engine. Export audios and contact transcripts will be available soon.

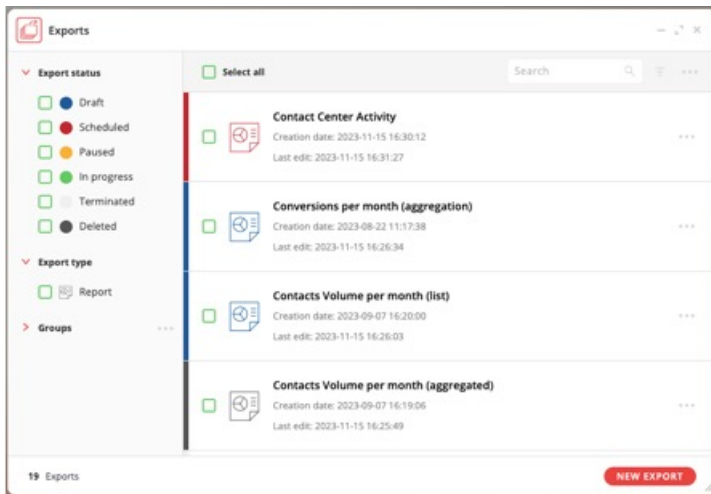


Figure 17.5. CoreMedia Export App

The exports app allows you to:

- List all performed exports
- Set a schedule for the export process (one time, every hour, daily, ... and custom)
- Specify the destination for the exported information (email, sftp or CoreMedia notification)
- Define the information to be exported
- Name and format (CSV, ODS, XLS and XLSX) of the file that will be exported

- Type of data (list values or aggregation values)
- Data source (contacts, IVRs, visitors, campaigns, leads, ...)
- Specific rules to filter information (fields change according to data source choose)
- Data (columns or metrics)
- Period (today, yesterday, current week, last week, last 7 days, ... and custom)
- Track what has been exported and what is planned for export
- Analyze the results of the export

### 17.1.9 CoreMedia Custom App

From an architectural point of view, CoreMedia allows you to quickly add functionality without changing the core of the platform by adding CoreMedia Apps. These apps have access to and can use CoreMedia data, create new visualizations and reports, extend existing segmentation functionality and provide new windows/widgets. The difference between CoreMedia Apps and CoreMedia Custom Apps is only in their ownership, the former being provided by CoreMedia while the latter can be implemented by any partner/customer.

In short, CoreMedia Custom Apps allows the creation of tailor-made applications, specifically designed to meet the unique needs of our customers and perfectly integrated into CoreMedia's backoffice environment.

### 17.1.10 CoreMedia DataSources

In our quest to provide unparalleled customer engagement, CoreMedia introduces the concept of DataSources. This innovative feature bridges the gap between CoreMedia's visitor profiles and a client's internal systems, allowing for dynamic enrichment of visitor data with external information.

A DataSource in CoreMedia is a powerful integration point that connects the visitor profiles on our platform with external data sources from client systems. This connection enables the import and synchronization of relevant data into the CoreMedia environment, offering a more comprehensive view of each visitor.

DataSources operate through two primary mechanisms:

- **Adcenter:** Clients can import data into CoreMedia contact lists, from which the platform reads and integrates information into visitor profiles. This method is ideal for batch updates and maintaining historical data consistency.

- **XML HTTP:** For real-time data synchronization, the XML HTTP DataSource fetches data from external systems via HTTP. This ensures that visitor profiles reflect the most current information from the client's systems.

### Update Mechanisms

DataSources offer two distinct update mechanisms to suit various use cases:

- **Update Value Only When Key Changes:** This method updates the visitor profile data only when there is a change in the identifying key, ensuring data consistency over time.
- **Update Value Only When Field is Shown:** Optimized for dynamic scenarios, this option updates data only when a specific field is displayed, providing real-time data relevancy.

By integrating DataSources, CoreMedia enables businesses to leverage their internal data assets for enhanced visitor segmentation and more personalized customer engagement strategies, thereby elevating the overall effectiveness of their digital presence.

## 17.2 CoreMedia Tag and Cookies

This document aims to provide an overview of the CoreMedia Tag and its behavior, as well as CoreMedia cookies.

### 17.2.1 CoreMedia Tag

**CoreMedia's Active TAG is the foundations for all interactions between CoreMedia's services and the website.**

Its primary function is to activate CoreMedia's services on the page where it is placed. It is responsible for collecting the necessary data to create visitor profiles, gather statistics, and enable the insertion of dynamic content (campaigns) configured in CoreMedia's back office based on customer-defined rules (segmentations).

To ensure the TAG remains static, all required functionalities, including the function library for various usage scenarios, are integrated into the TAG itself. This approach helps minimize the number of HTTP requests for loading additional modules and takes advantage of browser caching capabilities, prioritizing page loading and enhancing the user experience.

As a general principle, any HTTP requests necessary for loading dynamic content are launched only after the page's onLoad event, which ensures the basic page elements are loaded first. Additionally, CoreMedia's content is normally loaded asynchronously, in a non-blocking manner, allowing the loading of content from other sources and the execution of other JavaScript code during this process.

The CoreMedia TAG, similar to analytics tags like Google Analytics (GA), is specifically designed to collect navigation data. While it is commonly categorized as 'performance,' it's important to recognize that clients may have different perspectives and requirements, with some considering it as 'strictly necessary' or 'technical' based on their specific needs.

CoreMedia's TAG is common to all CoreMedia's customers (cannot be customized) and its usage is general for the entire service

General characteristics:

- Unique, static script;
- All requests are asynchronous and do not interfere in page load time;
- It is not affected by "3rd party cookie" filters;
- Size (compressed): about 28.2kB (in version v20230402a);

- As it is static, the visitor's browser will normally "cache" the TAG (it is called only once per visitor).

## 17.2.2 CoreMedia Cookies

Cookies are important in CoreMedia because they enable personalization, ad targeting, measurement and analytics, retargeting, and ad control. They allow marketers to deliver personalized experiences, target ads to specific audiences, measure campaign performance, retarget users, and manage ad frequency. However, cookie usage must comply with privacy regulations and user consent requirements.

According to GDPR, no cookie or similar mechanism can be created on a first visit to a website without explicit consent from the visitor. Thus, the most common cookie acceptance campaigns that do not block browsing are only GDPR compliant if no cookies are created until the visitor accepts the cookie policy.

However, different "cookie levels" are defined that differentiate the levels of tracking and data collection. These levels are often defined based on their impact on user privacy and the level of consent required for their use.

### 17.2.2.1 Cookie Levels

In addition to the global GDPR privacy rules, the following classifications were established in 2012 with the introduction of European e-Privacy directives, which were adapted in Portugal through Law 46/2020 that has been in force since August 30 and can be consulted in "Diário da República", (Article 5):

- **Strictly necessary cookies (or technical cookies)** These cookies enable the proper technical functioning of the website, ensure secure navigation, and allow for preferences to be remembered so that it is not necessary to reconfigure them each time you visit the site. Given these purposes, the collection of these cookies is mandatory.
- **Performance cookies** These cookies are used to obtain aggregated data on website usage for statistical purposes in order to continuously analyze and improve our service. For example, analytical cookies are used to know the daily volume of visits to the site or the volume of orders placed.
- **Personalization cookies** These cookies allow us to present recommendations and offers suitable to your interests on our website.
- **Advertising cookies** These cookies are from external entities (third parties) and allow for the presentation of ads that are relevant according to your interests on their respective sites. They also allow for the limitation of the

number of times you see a particular ad. These entities may also use this information for their own purposes. To find out which entities are involved, please refer to the Cookie Policy.

Please be aware that the provided descriptions may vary depending on the specific context and are solely intended for illustrative purposes.

## 17.2.2.2 Service Levels and CoreMedia Cookies

CoreMedia provides the following four levels of cookie classification:

- DO\_NOT\_TRACK
- FUNCTIONAL
- PERSONALIZATION
- TARGETING

## 17.2.3 CoreMedia Service

### 17.2.3.1 Operation Mode (Opt-In / Opt-Out)

By default, CoreMedia service operates in "Opt-In" mode. In this mode, a "first-party" cookie ("byside\_webcare\_tuid") is created on the client's domain with a visitor identifier that, during the lifetime of the cookie, allows the visitor to be recognized between visits.

If desired, CoreMedia can change this behavior and initiate sessions in "Opt-Out" mode (for example: session start until the client accepts the cookie policy). On each new visit, the visitor will be counted as a new visitor. To start the service in this mode, the CoreMedia tag must be loaded with the parameter "bysideWebcare\_privacy = 1".

Opt Out	Opt In
"BySide Privacy = 1 Cookie Level Service = Required"	"BySide Privacy = 0 Cookie Level Service = Targeting"

Opt Out	Opt In
"All campaigns will be presented <b>without</b> segmentation or intelligence"	"All campaigns will be presented <b>with</b> segmentation or intelligence"

Table 17.1. Table of error codes and Messages

```
<script>
  var bysideWebcare_webcare_id = "A82B12DB68";
  var bysideWebcare_lang = "en";
  var bysideWebcare_privacy = 1;
</script>

<script src="//www.clientdomain.com/assets/external/byside_webcare.js"
type="text/javascript"></script>
```

Example 17.2.

The method used to set cookies for new visitors ("var bysideWebcare\_privacy = 1;") only impacts the creation of the visitor. After this moment, changing the cookie level is only possible through the function mentioned here ("bysideWebcare\_setDoNotTrack").

Once the client accepts cookies, the following function must be called to allow the client to move to the respective cookie acceptance level (analytics):

```
bysideWebcare_setCookieServiceLevel( 'LEVEL',
  false, function () \{ bysideWebcare_reloadAgentContent(); } )
```

Example 17.3.

The value [LEVEL] can be one of the following:

- DO\_NOT\_TRACK;
- FUNCTIONAL;
- PERSONALIZATION;
- TARGETING.



# 17.2.4 CoreMedia Campaigns and Tag Management

## 17.2.4.1 Validate visitor type

Onsite campaigns have a configuration available that allows restricting the presentation of elements by visitor type. As shown in the image below, it is possible to present the campaign only to anonymous visitor type, only to identified visitor type, or by default, to all visitors.

As anonymous visitors, we include visitors at the **DO\_NOT\_TRACK** level.

Visitors who have a **FUNCTIONAL**, **PERSONALIZATION** or **TARGETING** service level are included in the identified visitor type.

## Segmentation Rules

The segmentation rules engine for campaigns includes a "Cookie Service Level" rule that allows you to limit the display of the campaign to visitors who have set the service level indicated in the rule

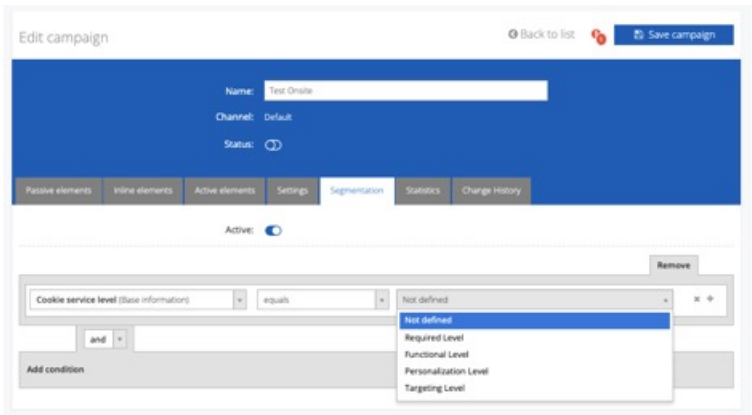


Figure 17.6.

## Setting service level via CoreMedia TAG Manager

When creating a TAG, it is possible to define the associated service level. The TAG will only be displayed to visitors who have defined the service level.

Figure 17.7.

### 17.2.4.2 Strategy for Increasing Opt-In Level

The CoreMedia service can, according to the privacy rules defined by each client, be used as a strategy to increase Opt-In levels. Regardless of the initial Opt-In level, the presentation of a Chat or Click2Call service, as an offer of contact/support with the customer, may request personal data (e.g. email and name) and require explicit acceptance of the "terms and conditions".

The content of the "terms and conditions", duly validated with the client's legal department, should make it clear that by accepting the "terms and conditions", the visitor/customer will automatically be accepting the respective level of Cookies (which may vary according to the CoreMedia client's privacy policy).

In this case, after acceptance of the "terms and conditions", the CoreMedia service may automatically:

1. Call the CoreMedia function to set the cookie level:

```
javascript bysideWebcare_setCookieServiceLevel( 'LEVEL', false, function ()
{ bysideWebcare_reloadAgentContent(); } )
```

2. Call CoreMedia client's cookie management solution to notify the change in cookie level of the visitor (more information in the next chapter).

CoreMedia service can thus be used as a strategy to increase Opt-In levels and manage first-party data of the client itself.

## 17.2.5 CoreMedia and Third-party Cookies Management

CoreMedia offers services for complete Cookie Management, allowing:

- Presenting widgets and cookie management campaigns;
- Managing third-party tags loading according to the Cookie level selected by the visitor.

Examples:



Figure 17.8.



Figure 17.9.

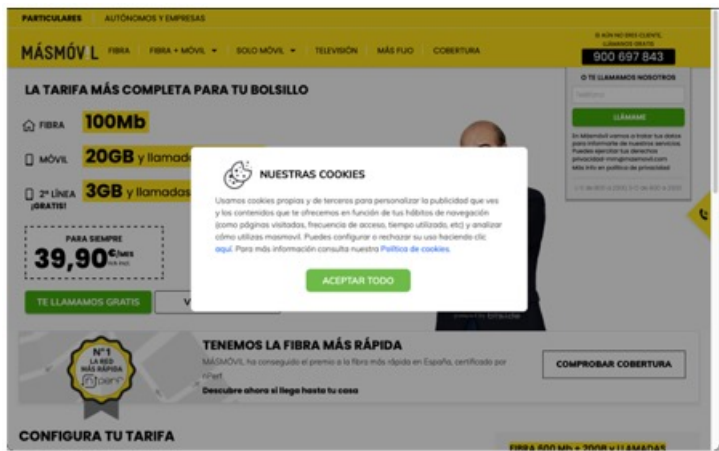


Figure 17.10.

# 18. Documentation and Training

## Empower Your Team with CoreMedia Enablement

In today's fast-paced digital landscape, staying ahead requires more than just great software—it requires the right expertise to maximize its potential. CoreMedia Enablement is a modern and innovative learning platform designed to equip your team with the knowledge and skills necessary to deliver high-end projects efficiently and confidently.

### A Smarter Way to Learn

CoreMedia Enablement offers a flexible, effective, and efficient approach to software education. Whether you're an architect, developer, administrator, or editor, our training programs cater to different roles, learning styles, and project needs.

- **Flexibility**

Access courses anytime, anywhere, ensuring your team can learn at their own pace while still benefiting from expert instruction and support.

- **Effectiveness**

Our rich and engaging learning environment fosters collaboration, communication, and critical thinking skills, enhancing overall knowledge retention.

- **Efficiency**

Save time and resources with structured training paths that streamline the learning process while allowing coaches to monitor progress and provide targeted feedback.

### Training Tailored to Your Needs

Our enablement program covers all aspects of CoreMedia's capabilities, offering courses for various roles and skill levels. Here are just a few examples:

- **CoreMedia Foundation**

A comprehensive introduction to the CoreMedia platform, covering all key components essential for implementation.

- **Frontend Development**

Hands-on training in creating themes with modern frameworks like Bootstrap, utilizing Freemarker templates, SASS, and JavaScript.

- **Content Application Engineering**

Deep dive into building and extending a CoreMedia content application engine (CAE).

- **Headless Delivery Developer**

Learn how to leverage CoreMedia's headless capabilities to integrate with various digital experience platforms.

- **Studio Customization**

Understand how to configure and tailor CoreMedia Studio to fit your organization's unique needs.

## CoreMedia Certification Program

For professionals looking to validate their expertise, the CoreMedia Certification Program provides an opportunity to earn industry recognized credentials. Certification tracks include:

- **Certified Frontend Developer**
- **Certified Delivery Developer**
- **Certified Management Developer**
- **Certified Headless Delivery Developer**
- **Certified Administrator**

## Enablement Plans

To support different learning needs, CoreMedia offers flexible enablement plans:

- **Flat S – 5,000€ / year**
- **Flat M – 10,000€ / year**
- **Flat L – 20,000€ / year**

## Documentation

Comprehensive documentation about all our products and components is available as PDF manuals as well as online documentation at <https://documentation.coremedia.com>.