

eXo Platform 3.0 - Content Functions

Copyright ©

1. Preface	1
1.1. Get Started with eXo Content	1
1.2. Package	1
2. Applications	3
2.1. Portlets	3
2.1.1. Content Detail	3
2.1.2. Content List	5
2.1.3. Search	9
2.1.4. Content Explorer	10
2.1.5. Administration	13
2.1.6. Fast Content Creator	14
2.1.7. Form Builder	16
2.1.8. Authoring	17
2.1.9. Newsletter	17
2.1.10. SEO portlet	18
3. Configuration	19
3.1. Components	19
3.1.1. ActionServiceContainer	19
3.1.2. ApplicationTemplateManagerService	19
3.1.3. FragmentCacheService	20
3.1.4. JodConverterService	20
3.1.5. LiveLinkManagerService	21
3.1.6. LockService	22
3.1.7. NewsletterInitializationService	22
3.1.8. NewsletterManagerService	25
3.1.9. SiteSearchService	25
3.1.10. SEOService	26
3.1.11. QueryService	28
3.1.12. TaxonomyService	28
3.1.13. ThumbnailService	30
3.1.14. TimelineService	31
3.1.15. WatchDocumentService	31
3.1.16. WCMSERVICE	32
3.2. External Component Plugins	33
3.2.1. AuthoringPublicationPlugin	33
3.2.2. BaseActionPlugin	33
3.2.3. BPActionPlugin	33
3.2.4. ContentTypeFilterPlugin	36
3.2.5. ContextPlugin	37
3.2.6. CreatePortalPlugin	38
3.2.7. ExcludeIncludeDataTypePlugin	38
3.2.8. FriendlyPlugin	39
3.2.9. ImageThumbnailPlugin	40
3.2.10. InitialWebcontentPlugin	42
3.2.11. LinkDeploymentPlugin	43
3.2.12. LockGroupsOrUsersPlugin	44
3.2.13. ManageDrivePlugin	45
3.2.14. ManageViewPlugin	49
3.2.15. PDFThumbnailPlugin	51
3.2.16. PorletTemplatePlugin	52
3.2.17. PredefinedProcessesPlugin	53
3.2.18. PublicationPlugin	54

3.2.19. QueryPlugin	55
3.2.20. RemovePortalPlugin	57
3.2.21. RemoveTaxonomyPlugin	57
3.2.22. ScriptActionPlugin	57
3.2.23. ScriptPlugin	60
3.2.24. StageAndVersionPublicationPlugin	63
3.2.25. StatesLifecyclePlugin	63
3.2.26. TagPermissionPlugin	65
3.2.27. TagStylePlugin	66
3.2.28. TaxonomyPlugin	69
3.2.29. TemplatePlugin	71
3.2.30. XMLdeploymentPlugin	75
4. Developer references	78
4.1. WCM Templates	78
4.1.1. Content types	78
4.1.2. List of Contents	88
4.2. WCM Explorer	89
4.2.1. CSS	89
4.2.2. CKEditor	89
4.3. Extensions	90
4.3.1. REST Services	90
4.3.2. UI Extensions	91
4.3.3. Authoring Extension	96
4.4. Public REST APIs	105
4.4.1. ThumbnailRESTService	105
4.4.2. RssConnector	106
4.4.3. FCKCoreRESTConnector	107
4.4.4. ResourceBundleConnector	108
4.4.5. VoteConnector	109
4.4.6. DriverConnector	110
4.4.7. GadgetConnector	111
4.4.8. PortalLinkConnector	112
4.4.9. GetEditedDocumentRESTService	112
4.4.10. PublicationGetDocumentRESTService	112
4.4.11. FavoriteRESTService	113
4.4.12. RESTImagesRendererService	114
4.4.13. LifecycleConnector	114
4.4.14. CopyContentFile	116
4.4.15. PDFViewerRESTService	116
4.5. Public Java APIs	117
4.5.1. TaxonomyService	117
4.5.2. LinkManager	122
4.5.3. PublicationManager	124
4.5.4. WCMComposer	125
4.5.5. NewFolksonomy	126
4.5.6. ApplicationTemplateManager	132
4.5.7. NodeFinder	134
4.5.8. JodConverter	136
4.5.9. SiteSearchService	137
4.5.10. SEOService	137
4.6. FAQ	139
4.6.1. How to deploy a workflow?	139

Chapter 1. Preface

1.1. Get Started with eXo Content

eXo Content is a fully integrated content management solution inside eXo Platform. Basically, in eXo Platform, the extension mechanism is used to add content features. To have more information about the extension mechanism, refer to the [GateIn reference guide](#).

This integrated solution provides users with a comprehensive platform about integrating applications, managing and publishing content - all from a single, familiar console.

When starting a new project, you can see eXo Content as a Java developer platform.

This document will give you guidelines related to building stable applications using eXo Content from the inside. The document consists of the following main contents:

- [Application](#) describes portlet applications included in eXo Content.
- [Configuration](#) provides you the comprehensive knowledge about the configuration with a large range of configuration parameters declared in eXo Content.
- [Developer References](#) describes about extension, public APIs, Java APIs, FAQs and related others in eXo Content.

1.2. Package

All package actions in eXo Content are started from the *delivery* folder, so you should go to this folder first.

```
$ cd ecms/delivery
```

- Package WCM standalone version - Tomcat bundle

```
$ cd wcm/assembly  
$ mvn clean install
```

- Package WCM with workflow enabled - Tomcat bundle

```
$ cd wkf-wcm/assembly  
$ mvn clean install
```

- Make WCM EAR packages to run with jBoss

1. Make WCM extension EAR

```
$ cd packaging/wcm/ear  
$ mvn clean install
```

2. Make WCM demo sites EAR

```
$ cd packaging/ecmdemo/ear  
$ mvn clean install
```

3. Make workflow EAR

```
$ cd packaging/workflow/ear  
$ mvn clean install
```

Chapter 2. Applications

This chapter provides you an comprehensive view about portlet applications included in eXo Content. These portlet applications are packaged as Web application archives (WARs).

Also, you can specify the package of each portlet and its available preferences that allow you to extend the configuration choices for standard preferences defined in *portlet.xml*.

2.1. Portlets

2.1.1. Content Detail

The **Content Detail** portlet allows users to view the detail of a specific content.

This is an example of the **Content Detail** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *presentation.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.scv.UISingleContentViewerPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
repository	String	repository	The repository where data are stored and maintained.
workspace	String	collaboration	The workspace where content is stored.
nodeIdentifier	String	N/A	The UUID or the path of content that you want to show.
ShowTitle	Boolean	true	Show the content title on the top of the portlet.
ShowDate	Boolean	false	Show the content date on the top of the portlet.
ShowOptionBar	Boolean	false	Show a bar with some actions (Print, Back).
ContextEnable	Boolean	false	Define if the portlet will use the parameter on URL as the path to content to display or not.
ParameterName	String	content-id	Define which parameter will be used to get the

Preference	Type	Value	Description
			content's path.
ShowVote	Boolean	false	Show the result of voting for the displayed content.
ShowComments	Boolean	false	Show the existing comments of this content (if any).
sharedCache	Boolean	true	Define if the portlet will cache the displayed contents.

- **Sample configuration**

```

<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>nodeIdentifier</name>
    <value>/myfolder/mycontent</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>ShowTitle</name>
    <value>>true</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>ShowDate</name>
    <value>>false</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>ShowOptionBar</name>
    <value>>false</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>ShowPrintAction</name>
    <value>>true</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>isQuickCreate</name>
    <value>>false</value>
    <read-only>>true</read-only>
  </preference>
  <preference>
    <name>ContextEnable</name>
    <value>>false</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>ParameterName</name>
    <value>content-id</value>
    <read-only>>false</read-only>
  </preference>
</portlet-preferences>

```

```
<name>sharedCache</name>
<value>true</value>
<read-only>false</read-only>
</preference>
</portlet-preferences>
```

Note

In WCM 2.3.0, some preferences are no longer used, for example, the ShowPrintAction preference.

2.1.2. Content List

The **Content List** portlet shows a list of contents which already exist in the system.

This is an example of the **Content List** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *presentation.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.clv.UICLVPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
mode	String	AutoViewerMode	The mode for displaying content of the portlet: all contents in a specific folder or all specific contents in the portlet.
folderPath	String		The path to the folder whose contents are displayed by this portlet.
orderBy	String	publication:liveDate	The property by which all the contents in the portlet are sorted.
orderType	String	DESC	The type of the content sort method: ascending or descending.
header	String		The header of the portlet which is displayed at the top of the portlet.
automaticDetection	Boolean	true	This value indicates whether the header of the portlet is selected to be the title of the folder given in the <i>folderPath</i> parameter (true value) or

Preference	Type	Value	Description
			the value given in the <i>header</i> parameter above.
formViewTemplatePath	String	/exo:ecm/views/templates/portal/1-the-templates-list/UIContent	The path to the template used to display the contents in this portlet.
paginatorTemplatePath	String	/exo:ecm/views/templates/portal/1-the-templates-list/paginators/	The path to the paginator used to display the contents in this portlet.
itemsPerPage	Integer	10	The number of contents displayed in every "page" of the portlet.
showThumbnailsView	Boolean	true	This value indicates whether the content image in this portlet is shown or not.
showTitle	Boolean	true	This value indicates whether the content title in this portlet is shown or not.
showHeader	Boolean	true	This value indicates whether the content header in this portlet is shown or not.
showRefreshButton	Boolean	false	This value indicates whether the Refresh button is shown in this portlet or not.
showDateCreated	Boolean	true	This value indicates whether the content created date in this portlet is shown or not.
showReadmore	Boolean	true	This value indicates whether the Read more button is shown in every content of the portlet or not. After clicking this button, the user can read the whole text of the content.
showSummary	Boolean	true	This value indicates whether the content summary in this portlet is shown or not.
showLink	Boolean	true	If this value is true, the

Preference	Type	Value	Description
			header of every content is also the link to view this content fully. If the value is false, the header is considered as a simple text.
showRssLink	Boolean	true	Show the RSS link of this portlet.
basePath	String	detail	Show the page in which the full content is displayed when the user clicks to the Read more button.
contextualFolder	String	contextualDisable	Enable/disable the contextual mode of the portlet. If enabled, the portlet can take the folder path indicated in the URL to display contents.
showScvWith	String	content-id	The parameter name which shows the folder path in URL when the Read more button is clicked.
showClvBy	String	folder-id	The parameter name which shows the folder path in URL.
sharedCache	Boolean	true	Define if the portlet will cache the displayed contents.

• Sample Configuration

```

<portlet-preferences>
  <preference>
    <name>mode</name>
    <value>AutoViewerMode</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>folderPath</name>
    <value/>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>orderBy</name>
    <value>publication:liveDate</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>orderType</name>

```

```

    <value>DESC</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>header</name>
    <value/>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>automaticDetection</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>formViewTemplatePath</name>
    <value>/exo:ecm/views/templates/content-list-viewer/list/UIContentListPresentationDefault.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>paginatorTemplatePath</name>
    <value>/exo:ecm/views/templates/content-list-viewer/paginators/UIPaginatorDefault.gtmpl</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>itemsPerPage</name>
    <value>10</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showThumbnailsView</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showTitle</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showHeader</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showRefreshButton</name>
    <value>false</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showDateCreated</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showReadmore</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showSummary</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showLink</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>showRssLink</name>
    <value>true</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>basePath</name>
    <value>detail</value>
    <read-only>false</read-only>
  </preference>
  <preference>

```

```

<name>contextualFolder</name>
<value>contextualDisable</value>
<read-only>>false</read-only>
</preference>
<preference>
  <name>showScvWith</name>
  <value>content-id</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>showClvBy</name>
  <value>folder-id</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>sharedCache</name>
  <value>>true</value>
  <read-only>>false</read-only>
</preference>
</portlet-preferences>

```

2.1.3. Search

The **Search** portlet allows users to do a search with any string. In eXo Content, there are three types of search: quick search, advanced search and search with saved queries.

The users can find this portlet in the front page. This is an example of the **Search** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *searches.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.search.UIWCMSearchPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
repository	string	repository	The place where data are stored and maintained.
workspace	string	collaboration	The workspace where the content is stored.
searchFormTemplatePath	string	/exo:ecm/views/templates/WTM path to the search form template. Advance Search/search-form/UIDefaultSearchForm.gtmpl	
searchResultTemplatePath	string	/exo:ecm/views/templates/WTM path to the search result template. Advance Search/search-result/UIDefaultSearchResult.gtmpl	
searchPaginatorTemplatePath	string	/exo:ecm/views/templates/WTM path to the search paginator template. Advance Search/search-paginator/UIDefaultSearchPaginator.gtmpl	
searchPageLayoutTemplatePath	string	/exo:ecm/views/templates/WTM path to the search page template. Advance Search/search-page-layout/UISearchPageLayoutDefault.gtmpl	

Preference	Type	Value	Description
itemsPerPage	Integer	5	The number of items for each page.
showQuickEditButton	boolean	true	Show or hide the quick edit icon.
basePath	string	parameterizedviewer	The page which is used to display the search result.

- **Sample configuration**

```

<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>searchFormTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-form/UIDefaultSearchForm.gtmpl</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>searchResultTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-result/UIDefaultSearchResult.gtmpl</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>searchPaginatorTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-paginator/UIDefaultSearchPaginator.gtmpl</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>searchPageLayoutTemplatePath</name>
    <value>/exo:ecm/views/templates/WCM Advance Search/search-page-layout/UISearchPageLayoutDefault.gtmpl</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>itemsPerPage</name>
    <value>5</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>showQuickEditButton</name>
    <value>true</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>basePath</name>
    <value>parameterizedviewer</value>
    <read-only>>false</read-only>
  </preference>
</portlet-preferences>

```

2.1.4. Content Explorer

The **Content Explorer** portlet is used to manage all documents in different drives. With this portlet, users can

do many different actions depending on their roles, such as adding/deleting a category and a document, showing/hiding a node, managing publication, and more.

This is an example of the **Content Explorer** portlet used in eXo Content:

- **Packaging:** The portlet is packaged in the *ecmexplorer.war* file.
- **The portlet class name:** *org.exoplatform.ecm.webui.component.explorer.UIJCRExplorerPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
repository	string	repository	The repository name which is used in an instance of Content Explorer.
workspace	string	N/A	Not in use. The workspace name was included in the Drive.
path	string	N/A	The path of the node. This preference will be used when the selected usecase is Parameterize .
drive	string	N/A	Not in use. Replaced by the <code>driveName</code> preference.
views	string	N/A	Not in use. The views will be displayed basing on the Drive which the user has the access permission.
allowCreateFolders	string	N/A	Allow creating a folder by type. When you do not specify the value, the default value will be <code>nt:unstructured, nt:folder</code> .
categoryMandatoryWhenFileUpload	boolean	false	Force a user to add a category when uploading or creating a document.
uploadFileSizeLimitMB	float	150	The maximum size of a file that is uploaded to the system (MB).
usecase	string	selection	The behavior to access Content Explorer. By default, the "selection" option is configured.

Preference	Type	Value	Description
			Besides "selection", there are four other ways to configure the Content Explorer: Jailed, Personal, Social, Parameterize.
driveName	string	private	The name of drive which the user wants to access.
trashHomeNodePath	string	/Trash	The location to store the deleted nodes.
trashRepository	string	repository	The name of the repository where stores the deleted nodes.
trashWorkspace	string	collaboration	The name of the workspace where stores the deleted nodes.
editInNewWindow	boolean	false	Allow editing documents with or without a window popup.
showTopBar	boolean	true	Allow showing the Top bar or not.
showActionBar	boolean	true	Allow showing the Action bar or not.
showSideBar	boolean	true	Allow showing the Side bar or not.
showFilterBar	boolean	true	Allow showing the Filter bar or not.

• Sample Configuration

```

<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value/>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>path</name>
    <value/>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>drive</name>
    <value/>
    <read-only>>false</read-only>

```

```

</preference>
<preference>
  <name>views</name>
  <value/>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>allowCreateFolders</name>
  <value/>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>categoryMandatoryWhenFileUpload</name>
  <value>>false</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>uploadFileSizeLimitMB</name>
  <value>150</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>usecase</name>
  <value>selection</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>driveName</name>
  <value>Private</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>trashHomeNodePath</name>
  <value>/Trash</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>trashRepository</name>
  <value>repository</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>trashWorkspace</name>
  <value>collaboration</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>editInNewWindow</name>
  <value>>false</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>showTopBar</name>
  <value>>true</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>showActionBar</name>
  <value>>true</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>showSideBar</name>
  <value>>true</value>
  <read-only>>false</read-only>
</preference>
<preference>
  <name>showFilterBar</name>
  <value>>true</value>
  <read-only>>false</read-only>
</preference>
</portlet-preferences>

```

2.1.5. Administration

The **Administration** portlet is used to manage the main ECM functions, including categories and tags, content presentation, types of content and advanced configuration.

This is an example of the **Administration** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *ecmadmin.war* file.
- **The portlet class name:** *org.exoplatform.ecm.webui.component.admin.UIECMAdminPortlet*
- **Available preferences:** When using this portlet, you can customize the following preference:

Preference	Type	Value	Description
repository	string	Repository	The name of the current repository.

- **Sample Configuration**

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

2.1.6. Fast Content Creator

The **Fast Content Creator** portlet consists of two modes: **Standard Content Creator** and **Basic Content Creator**. This portlet allows users to quickly create contents without accessing the Content Explorer portlet.

This is an example of the **Fast Content Creator** portlet used in eXo Content:

By default, this portlet is applied for the Contact Us portlet in eXo Content.

- **Packaging:** This portlet is packaged in the *formgenerator.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.fastcontentcreator.UIFCCPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
mode	string	basic	The default mode of the Fast Content Creator portlet.

Preference	Type	Value	Description
repository	string	repository	The name of the current repository.
workspace	string	collaboration	The workspace where the content is stored.
path	string	/Groups/platform/users/Documents	The destination path where the content is stored.
type	string	exo:article	The node type of document which is shown on the dialog form.
saveButton	string	Save	The custom button: Save .
saveMessage	string	This node has been saved successfully	The custom message when the user clicks the Save button.
isRedirect	boolean	false	Specify whether redirecting to another page or not.
redirectPath	string	http://www.google.com.vn	The path to which the page will redirect.
isActionNeeded	boolean	true	Specify whether an action is needed to save to the configuration or not.

• Sample Configuration

```

<portlet-preferences>
  <preference>
    <name>mode</name>
    <value>basic</value>
    <read-only>true</read-only>
  </preference>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>path</name>
    <value>/Groups/platform/users/Documents</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>type</name>
    <value>exo:article</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>saveButton</name>

```

```

    <value>Save</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>saveMessage</name>
    <value>This node has been saved successfully</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>isRedirect</name>
    <value>>false</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>redirectPath</name>
    <value>http://www.google.com.vn</value>
    <read-only>>false</read-only>
  </preference>
  <preference>
    <name>isActionNeeded</name>
    <value>true</value>
    <read-only>true</read-only>
  </preference>
</portlet-preferences>

```

2.1.7. Form Builder

Warning

The Form Builder portlet and its services are deprecated. It remains fully supported for eXo customers, however it will not receive any enhancement and will be removed from the product scope in the future.

The **Form Builder** portlet allows users to create node types and document templates for node types.

This is an example of the **Form Builder** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *formgenerator.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.formgenerator.UIFormGeneratorPortlet*
- **Available preferences:** When using this portlet, you can customize the following preference:

Preference	Type	Value	Description
repository	string	repository	The current repository name which is always "repository".

- **Sample Configuration**

```

<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>>false</read-only>
  </preference>

```

```
</portlet-preferences>
```

2.1.8. Authoring

The **Authoring** portlet allows users to manage contents in draft and ones which need to be approved or published.

This is an example of the **Authoring** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *authoring-apps.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.authoring.UWCMDashboardPortlet*
- **Available preferences:** When using this portlet, you can customize the following preferences:

Preference	Type	Value	Description
repository	string	Repository	The name of the repository.
workspace	string	Collaboration	The name of the workspace.
drive	string	Collaboration	The name of the drive.

- **Sample Configuration**

```
<portlet-preferences>
  <preference>
    <name>repository</name>
    <value>repository</value>
    <read-only>true</read-only>
  </preference>
  <preference>
    <name>workspace</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
  <preference>
    <name>drive</name>
    <value>collaboration</value>
    <read-only>false</read-only>
  </preference>
</portlet-preferences>
```

2.1.9. Newsletter

Warning

The Newsletter portlet and its services are deprecated. It remains fully supported for eXo customers, however it will not receive any enhancement and will be removed from the product scope in the future.

The **Newsletter** portlet is used to help users quickly get the updated newsletter from a site.

This is an example of the **Newsletter** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *newsletter.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.newsletter.manager.UINewsletterManagerPortlet*

2.1.10. SEO portlet

The **SEO** portlet allows users to manage SEO data of web content and web pages, so they can maximize their website position on search engines.

This is an example of the **SEO** portlet used in eXo Content:

- **Packaging:** This portlet is packaged in the *seo.war* file.
- **The portlet class name:** *org.exoplatform.wcm.webui.seo.UISEOToolbarPortlet*

Chapter 3. Configuration

This chapter describes about configuration used in eXo Content. It consists of two main sections:

- [Component](#)
- [External Component Plugins](#)

3.1. Components

This section describes services which provide low-level functionality for UI components.

3.1.1. ActionServiceContainer

The *ActionServiceContainer* component is used to manage actions (adding, removing, or executing actions, and more) in the system. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.actions.ActionServiceContainer</key>
  <type>org.exoplatform.services.cms.actions.impl.ActionServiceContainerImpl</type>
  <init-params>
    <value-param>
      <name>workspace</name>
      <value>system</value>
    </value-param>
    <value-param>
      <name>repository</name>
      <value>repository</value>
    </value-param>
  </init-params>
</component>
```

Details:

- **Value-param:**

Name	Type	Value	Description
workspace	string	system	The workspace name.
repository	string	repository	The repository name.

3.1.2. ApplicationTemplateManagerService

The *ApplicationTemplateManagerService* component is used to manage dynamic Groovy templates for WCM-based products. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.views.ApplicationTemplateManagerService</key>
  <type>org.exoplatform.services.cms.views.impl.ApplicationTemplateManagerServiceImpl</type>
```

```

<init-params>
  <properties-param>
    <name>storedLocations</name>
    <property name="repository" value="system"/>
  </properties-param>
</init-params>
</component>

```

Details:

- **Properties-param:**

Name	Property name	Type	Value	Description
storedLocations	repository	string	system	The repository name.

3.1.3. FragmentCacheService

The *FragmentCacheService* component is used to cache the response fragments which are sent to end-users.

The configuration of this component is found in *core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/wcm-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.portletcache.FragmentCacheService</key>
  <type>org.exoplatform.services.portletcache.FragmentCacheService</type>
  <init-params>
    <value-param>
      <name>cleanup-cache</name>
      <description>The cleanup cache period in seconds</description>
      <value>300</value>
    </value-param>
  </init-params>
</component>

```

Details

- **Value-param:**

Name	Type	Value	Description
cleanup-cache	integer	300	The time period over which cache items are expired.

3.1.4. JodConverterService

The *JodConverterServices* component is used to convert documents into different office formats. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.cms.jodconverter.JodConverterService</key>

```

```

<type>org.exoplatform.services.cms.jodconverter.impl.JodConverterServiceImpl</type>
<init-params>
  <value-param>
    <name>host</name>
    <value>127.0.0.1</value>
  </value-param>
  <value-param>
    <name>port</name>
    <value>8100</value>
  </value-param>
</init-params>
</component>

```

Details:

- **Value-param:**

Name	Type	Value	Description
host	The host IP	127.0.0.1	The host from which a document is converted into different office formats.
port	The port number	8100	The port number is open to accept converting a document into different office formats.

3.1.5. LiveLinkManagerService

The *LiveLinkManagerService* component is used to check broken links, update links when the links are edited and extract links to return a list of all links. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/wcm-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.wcm.link.LiveLinkManagerService</key>
  <type>org.exoplatform.services.wcm.link.LiveLinkManagerServiceImpl</type>
  <init-params>
    <properties-param>
      <name>server.config</name>
      <description>server.address</description>
      <property name="scheme" value="http"/>
      <property name="hostName" value="localhost"/>
      <property name="port" value="8080"/>
    </properties-param>
  </init-params>
</component>

```

Details:

Properties-param	Property name	Type	Value	Description
server.config	scheme	http/https	http	All the property names are used together to configure the server. Here is an
	hostName	String	localhost	

Properties-param	Property name	Type	Value	Description
	port	The port number	8080	example about the server configuration: http://localhost:8080.

3.1.6. LockService

The *LockService* component is used to manage all locked nodes and allows unlocking the locked nodes in the system. It is also used to assign the Lock right to a user or a user group or a membership. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.lock.LockService</key>
  <type>org.exoplatform.services.cms.lock.impl.LockServiceImpl</type>
</component>
```

3.1.7. NewsletterInitializationService

The *NewsletterInitializationService* component is used to initiate some data for the newsletter portlet. The configuration of this component is found in *packaging/ecmdemo/webapp/src/main/webapp/WEB-INF/conf/sample-portal/wcm/newsletter-configuration.xml*.

Warning

The Newsletter portlet and its services are deprecated. It remains fully supported for eXo customers, however it will not receive any enhancement and will be removed from the product scope in the future.

```
<component>
  <type>org.exoplatform.services.wcm.newsletter.NewsletterInitializationService</type>
  <init-params>
    <values-param>
      <name>portalNames</name>
      <value>classic</value>
      <value>acme</value>
    </values-param>
    <values-param>
      <name>administrators</name>
      <value>root</value>
      <value>john</value>
    </values-param>
    <object-param>
      <name>marketing</name>
      <description>marketing</description>
      <object type="org.exoplatform.services.wcm.newsletter.NewsletterCategoryConfig">
        <field name="name">
          <string>marketing</string>
        </field>
        <field name="title">
          <string>Marketing</string>
        </field>
        <field name="description">
          <string>You want to know where we are, where we go ?</string>
        </field>
        <field name="moderator">
```

```

        <string>*:/platform/web-contributors</string>
    </field>
</object>
</object-param>
<object-param>
    <name>general</name>
    <description>general</description>
    <object type="org.exoplatform.services.wcm.newsletter.NewsletterCategoryConfig">
        <field name="name">
            <string>general</string>
        </field>
        <field name="title">
            <string>General</string>
        </field>
        <field name="description">
            <string>General information about us</string>
        </field>
        <field name="moderator">
            <string>*:/platform/web-contributors</string>
        </field>
    </object>
</object-param>
<object-param>
    <name>subscription2</name>
    <description>subscription2</description>
    <object type="org.exoplatform.services.wcm.newsletter.NewsletterSubscriptionConfig">
        <field name="name">
            <string>results</string>
        </field>
        <field name="title">
            <string>Results</string>
        </field>
        <field name="description">
            <string>Monthly newsletter about our results and forecasts</string>
        </field>
        <field name="categoryName">
            <string>general</string>
        </field>
        <field name="redactor">
            <string>*:/platform/web-contributors</string>
        </field>
    </object>
</object-param>
<object-param>
    <name>subscription1</name>
    <description>subscription1</description>
    <object type="org.exoplatform.services.wcm.newsletter.NewsletterSubscriptionConfig">
        <field name="name">
            <string>checklist</string>
        </field>
        <field name="title">
            <string>Check-List</string>
        </field>
        <field name="description">
            <string>Weekly newsletter with general topics</string>
        </field>
        <field name="categoryName">
            <string>general</string>
        </field>
        <field name="redactor">
            <string>*:/platform/web-contributors</string>
        </field>
    </object>
</object-param>
<object-param>
    <name>subscription3</name>
    <description>subscription3</description>
    <object type="org.exoplatform.services.wcm.newsletter.NewsletterSubscriptionConfig">
        <field name="name">
            <string>market</string>
        </field>
        <field name="title">
            <string>The market</string>
        </field>
        <field name="description">
            <string>What's on the market today ?</string>
        </field>
        <field name="categoryName">
            <string>marketing</string>
        </field>
    </object>
</object-param>

```

```

        </field>
        <field name="redactor">
            <string>*:/platform/web-contributors</string>
        </field>
    </object>
</object-param>
<object-param>
    <name>user1@gmail.com</name>
    <description>user1@gmail.com</description>
    <object type="org.exoplatform.services.wcm.newsletter.config.NewsletterUserConfig">
        <field name="mail">
            <string>user1@gmail.com</string>
        </field>
    </object>
</object-param>
<object-param>
    <name>user2@gmail.com</name>
    <description>user2@gmail.com</description>
    <object type="org.exoplatform.services.wcm.newsletter.config.NewsletterUserConfig">
        <field name="mail">
            <string>user2@gmail.com</string>
        </field>
    </object>
</object-param>
</init-params>
</component>

```

Details:

- **Value-param**

Name	Type	Value	Description
portalNames	string	classic, acme	The portal names.
administrators	string	root	The administrator who manages the newsletter portlet.

- **Object-param:**

- **object type:** org.exoplatform.services.wcm.newsletter.NewsletterCategoryConfig

Field	Type	Description
name	string	The name of categories or subscriptions.
title	string	The title of categories or subscriptions.
description	string	The description of categories or subscriptions.
moderator	string	The users or groups that can moderate the newsletter portlet.
categoryName	string	The categories name.
redactor	string	The users or groups that are newsletter redactor.
mail	string	The email that user uses to

Field	Type	Description
		subscribe.

3.1.8. NewsletterManagerService

The *NewsletterManagerService* component is used to send newsletters to subscribers. The configuration of this component is found in *core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/ext-newsletter-configuration.xml*.

Warning

The Newsletter Manager portlet and its services are deprecated. It remains fully supported for eXo customers, however it will not receive any enhancement and will be removed from the product scope in the future.

```
<component>
  <type>org.exoplatform.services.wcm.newsletter.NewsletterManagerService</type>
  <init-params>
    <value-param>
      <name>repository</name>
      <value>repository</value>
    </value-param>
    <value-param>
      <name>workspace</name>
      <value>collaboration</value>
    </value-param>
  </init-params>
</component>
```

Details:

- **Value-param:**

Name	Type	Value	Description
repository	string	repository	The repository name.
workspace	string	collaboration	The workspace name.

3.1.9. SiteSearchService

The *SiteSearchService* component is used in the Search portlet that allows users to find all information matching with your given keyword.

It is configured in the *core/core-configuration/src/main/webapp/WEB-INF/conf/configuration.xml* file as follows:

```
<import>war:/conf/wcm-core/core-search-configuration.xml</import>
```

The component configuration maps the *SiteSearchService* component with its own implementation:

SiteSearchServiceImpl.

```

<component>
  <key>org.exoplatform.services.wcm.search.SiteSearchService</key>
  <type>org.exoplatform.services.wcm.search.SiteSearchServiceImpl</type>
  <component-plugins>
    ...
  </component-plugins>
  <init-params>
    <value-param>
      <name>isEnabledFuzzySearch</name>
      <value>true</value>
    </value-param>
    <value-param>
      <name>fuzzySearchIndex</name>
      <value/>
    </value-param>
  </init-params>
</component>

```

Detail:• **Value-param:**

Name	Type	Value	Description
isEnabledFuzzySearch	boolean	true	Allow administrators to enable/disable the fuzzy search mechanism.
fuzzySearchIndex	N/A	N/A	Allow the approximate level between the input keyword and the found key results. In case of the invalid configuration, the default value is set to 0.8.

To have more information about the fuzzy search, please refer to [Fuzzy Search](#).

3.1.10. SEOService

The *SEOService* component is used to help users manage SEO data of a page or a content, so their websites can achieve higher rankings on search engines. The configuration of this component is found in */packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/wcm-extension/wcm/seo-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.seo.SEOService</key>
  <type>org.exoplatform.services.seo.impl.SEOServiceImpl</type>
  <init-params>
    <object-param>
      <name>seo.config</name>
      <object type="org.exoplatform.services.seo.SEOConfig">
        <field name="robotsindex">
          <collection type="java.util.ArrayList">
            <value>
              <string>INDEX</string>
            </value>
            <value>
              <string>NOINDEX</string>
            </value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component>

```

```

</field>
<field name="robotsfollow">
  <collection type="java.util.ArrayList">
    <value>
      <string>FOLLOW</string>
    </value>
    <value>
      <string>NOFOLLOW</string>
    </value>
  </collection>
</field>
<field name="frequency">
  <collection type="java.util.ArrayList">
    <value>
      <string>Always</string>
    </value>
    <value>
      <string>Hourly</string>
    </value>
    <value>
      <string>Daily</string>
    </value>
    <value>
      <string>Weekly</string>
    </value>
    <value>
      <string>Monthly</string>
    </value>
    <value>
      <string>Yearly</string>
    </value>
    <value>
      <string>Never</string>
    </value>
  </collection>
</field>
</object>
</object-param>
</init-params>
</component>

```

Details:

- **Object-param:**
 - **Object type:** org.exoplatform.services.seo.SEOConfig

Field	Type	Value	Description
robotsindex	ArrayList	INDEX NOINDEX	Allow search engines to index a particular page or not.
robotsfollow	ArrayList	FOLLOW NOFOLLOW	Allow search engines to follow links from a particular page to find other pages or not.
frequency	ArrayList	Always Hourly Daily	Define how often a particular page is updated.
			27

Field	Type	Value	Description
		Weekly	
		Monthly	
		Yearly	
		Never	

3.1.11. QueryService

The *QueryService* component is used to manage many queries, including adding, removing or executing a query. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.queries.QueryService</key>
  <type>org.exoplatform.services.cms.queries.impl.QueryServiceImpl</type>
  <init-params>
    <value-param>
      <name>workspace</name>
      <value>system</value>
    </value-param>
    <value-param>
      <name>relativePath</name>
      <value>Private/Queries</value>
    </value-param>
    <value-param>
      <name>group</name>
      <value>*:/admin</value>
    </value-param>
  </init-params>
</component>
```

Details:

- **Value-param:**

Name	Type	Value	Description
workspace	string	system	The workspace name.
relativePath	Private/Queries	Private/Queries	The path to the query location.
group	string	*:/admin	The group is allowed to access the query folder.

3.1.12. TaxonomyService

The *TaxonomyService* component is used to sort documents to ease searches when browsing documents online. It provides a multi-dimensional set of paths to find a document. In many cases, you can get your content by

using different category paths. Therefore, after creating a document somewhere in the repository, it is possible to categorize it by adding several taxonomy references. By browsing the taxonomy tree, it will be possible to find the referencing article and display them as if they were children of the taxonomy nodes. Taxonomies are stored in the JCR itself and the JCR Reference functionality is used to provide that advanced WCM feature. The tree of taxonomies can be managed simply, such as copying/cutting/pasting nodes, or adding and removing taxonomies from the tree. Once a taxonomy has been added, any user who has access to the "Manage Categories" icon from his/her view can then browse the taxonomy tree and refer one of its nodes to the created documents.

```
<component>
  <key>org.exoplatform.services.cms.taxonomy.TaxonomyService</key>
  <type>org.exoplatform.services.cms.taxonomy.impl.TaxonomyServiceImpl</type>
  <init-params>
    <object-param>
      <name>defaultPermission.configuration</name>
      <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission">
        <field name="permissions">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPerm
                <field name="identity">
                  <string>*:/platform/administrators</string>
                </field>
                <field name="read">
                  <string>true</string>
                </field>
                <field name="addNode">
                  <string>true</string>
                </field>
                <field name="setProperty">
                  <string>true</string>
                </field>
                <field name="remove">
                  <string>true</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPerm
                <field name="identity">
                  <string>*:/platform/users</string>
                </field>
                <field name="read">
                  <string>true</string>
                </field>
                <field name="addNode">
                  <string>true</string>
                </field>
                <field name="setProperty">
                  <string>true</string>
                </field>
                <field name="remove">
                  <string>>false</string>
                </field>
              </object>
            </value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component>
```

Details:

- **Object type:** *org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission*

Field	Type	Value	Description
permissions	ArrayList	<i>{java.util.ArrayList}</i>	The list of the default user permissions to access the taxonomy tree.

- **Object**

type:

org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission\$Permission

Field	Type	Description
identity	string	The name of user, group or membership.
read	boolean	The permission to read the taxonomy tree.
addNode	boolean	The permission to add a node to the taxonomy tree.
setProperty	boolean	The permission to set properties for a node in the taxonomy tree.
remove	boolean	The permission to remove a node from the taxonomy tree.

3.1.13. ThumbnailService

The *ThumbnailService* component is used to resize all the images into different sizes. Besides the default sizes, it also allows users to customize the images into the desired sizes. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.thumbnail.ThumbnailService</key>
  <type>org.exoplatform.services.cms.thumbnail.impl.ThumbnailServiceImpl</type>
  <init-params>
    <value-param>
      <name>smallSize</name>
      <value>32x32</value>
    </value-param>
    <value-param>
      <name>mediumSize</name>
      <value>64x64</value>
    </value-param>
    <value-param>
      <name>largeSize</name>
      <value>300x300</value>
    </value-param>
    <value-param>
      <name>enable</name>
      <value>false</value>
    </value-param>
    <value-param>
      <name>mimetypes</name>
      <value>image/jpeg;image/png;image/gif;image/bmp</value>
    </value-param>
  </init-params>
</component>
```

Details:• **Value-param:**

Name	Type	Value	Description
smallSize	integer x integer	32x32	The small thumbnail size.
mediumSize	integer x integer	64x64	The medium thumbnail size.
largeSize	integer x integer	300x300	The large thumbnail size.
enable	boolean	false	Specify if the thumbnail is displayed or not.
mimetypes	Images formats	image/jpeg;image/png;image/gif;image/tiff	The image formats are supported.

3.1.14. TimelineService

The *TimelineService* component allows documents to be displayed by days, months and years. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```
<component>
  <key>org.exoplatform.services.cms.timeline.TimelineService</key>
  <type>org.exoplatform.services.cms.timeline.impl.TimelineServiceImpl</type>
  <init-params>
    <value-param>
      <name>itemPerTimeline</name>
      <value>5</value>
    </value-param>
  </init-params>
</component>
```

Details:• **Value-param**

Name	Type	Value	Description
itemPerTimeline	integer	5	The number of documents are displayed.

3.1.15. WatchDocumentService

The *WatchDocumentService* component allows users to watch/unwatch a document. If they are watching the document, they will receive a notification mail when there are any changes on the document. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.cms.watch.WatchDocumentService</key>
  <type>org.exoplatform.services.cms.watch.impl.WatchDocumentServiceImpl</type>
  <init-params>
    <object-param>
      <name>messageConfig</name>
      <description>description</description>
      <object type="org.exoplatform.services.cms.watch.impl.MessageConfig">
        <field name="sender">
          <string>support@exoplatform.com</string>
        </field>
        <field name="subject">
          <string>Your watching document is changed</string>
        </field>
        <field name="content">
          <string>The document that you are watching is changed.
            Please go to ecm to see this change
          </string>
        </field>
      </object>
    </object-param>
  </init-params>
</component>

```

Details:

- **object-param:**
 - **Object type:** *org.exoplatform.services.cms.watch.impl.MessageConfig*

Field	Type	Value	Description
sender	string	support@exoplatform.com	The sender who sends the notification mail.
subject	string	Your watching document is changed.	The subject of the notification mail.
content	string	The document that you are watching is changed. Please go to ecm to see this change.	The content of the notification mail.

3.1.16. WCMService

The *WCMService* component allows setting expiration cache of portlets and checking given portals if they are shared portals or not. It also gets reference contents basing on item identifiers. The configuration of this component is found in */core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml*.

```

<component>
  <key>org.exoplatform.services.wcm.core.WCMService</key>
  <type>org.exoplatform.services.wcm.core.impl.WCMServiceImpl</type>
  <init-params>
    <properties-param>
      <name>server.config</name>
      <description>server.config</description>
      <property name="expirationCache" value="30"/>
    </properties-param>
  </init-params>
</component>

```

Details:

Properties-param	Property name	Type	Value	Description
server.config	expirationCache	integer	30	The period in which the cache is clear in second. By default, the cache is cleared every 30 seconds.

3.2. External Component Plugins

This section describes about the main component plugins used in eXo Content. Each part supply an example configuration with the explanation about ini-params so you can know how to use these plugins.

3.2.1. AuthoringPublicationPlugin

This plugin is used to manage the publication lifecycle of web contents and DMS document on a portal page with more states and versions. The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/content-extended/authoring/configuration.xml`.

Sample configuration:

```
<component-plugin>
  <name>Authoring publication</name>
  <set-method>addPublicationPlugin</set-method>
  <type>org.exoplatform.services.wcm.extensions.publication.lifecycle.authoring.AuthoringPublicationPlugin</type>
  <description>This publication lifecycle publish a web content or DMS document to a portal page with more states and version.
</description>
</component-plugin>
```

In which:

- **Name:** *Authoring publication*
- **Set-method:** *addPublicationPlugin*
- **Type:** *org.exoplatform.services.wcm.extensions.publication.lifecycle.authoring.AuthoringPublicationPlugin*

3.2.2. BaseActionPlugin

This is an abstract class and the parent of [ScriptActionPlugin](#) and [BPActionPlugin](#) . You can create a link from [this plugin to its children](#).

3.2.3. BPActionPlugin

This plugin is used to define the business process action in Workflow.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.actions.ActionServiceContainer</target-component>
```

The configuration is applied mainly in `packaging/workflow/webapp/src/main/webapp/WEB-INF/workflow-extension/workflow/workflow-system-configuration.xml`

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.actions.ActionServiceContainer</target-component>
  <component-plugin>
    <name>exo:businessProcessAction</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.plugin.actions.impl.BPActionPlugin</type>
    <priority>112</priority>
    <init-params>
      <object-param>
        <name>predefined.actions</name>
        <description>description</description>
        <object type="org.exoplatform.services.cms.actions.impl.ActionConfig">
          <field name="repository">
            <string>repository</string>
          </field>
          <field name="workspace">
            <string>collaboration</string>
          </field>
          <field name="actions">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Action">
                  <field name="type">
                    <string>exo:publishingProcess</string>
                  </field>
                  <field name="name">
                    <string>content publishing</string>
                  </field>
                  <field name="description">
                    <string>content publishing workflow</string>
                  </field>
                  <field name="srcWorkspace">
                    <string>collaboration</string>
                  </field>
                  <field name="srcPath">
                    <string>/Documents/Validation Requests</string>
                  </field>
                  <field name="isDeep">
                    <boolean>true</boolean>
                  </field>
                  <field name="lifecyclePhase">
                    <collection type="java.util.ArrayList">
                      <value>
                        <string>node_added</string>
                      </value>
                    </collection>
                  </field>
                  <field name="roles">
                    <string>*:/platform/users</string>
                  </field>
                  <field name="variables">
                    <string>exo:supervisor=*:/organization/management/executive-board;exo:valida
                  </string>
                  </field>
                  <field name="mixins">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.actions.impl.ActionCo
                          <field name="name">
                            <string>exo:publishLocation</string>
                          </field>
                          <field name="properties">
```

```

        <string>
            exo:publishWorkspace=collaboration;exo:publishPath=/
        </string>
    </field>
</object>
</value>
<value>
    <object type="org.exoplatform.services.cms.actions.impl.ActionCo
        <field name="name">
            <string>exo:pendingLocation</string>
        </field>
        <field name="properties">
            <string>
                exo:pendingWorkspace=collaboration;exo:pendingPath=/
            </string>
        </field>
    </object>
</value>
<value>
    <object type="org.exoplatform.services.cms.actions.impl.ActionCo
        <field name="name">
            <string>exo:backupLocation</string>
        </field>
        <field name="properties">
            <string>exo:backupWorkspace=backup;exo:backupPath=/Expir
                Documents
            </string>
        </field>
    </object>
</value>
<value>
    <object type="org.exoplatform.services.cms.actions.impl.ActionCo
        <field name="name">
            <string>exo:trashLocation</string>
        </field>
        <field name="properties">
            <string>
                exo:trashWorkspace=collaboration;exo:trashPath=/Docu
            </string>
        </field>
    </object>
</value>
<value>
    <object type="org.exoplatform.services.cms.actions.impl.ActionCo
        <field name="name">
            <string>mix:affectedNodeTypes</string>
        </field>
        <field name="properties">
            <string>
                exo:affectedNodeTypeNames=exo:article,exo:podcast,ex
            </string>
        </field>
    </object>
</value>
</collection>
</field>
</object>
</value>
</collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** exo:businessProcessAction
- **Set-method:** addPlugin
- **Type:** org.exoplatform.services.plugin.actions.impl.BPActionPlugin

- **Object type:** `org.exoplatform.services.cms.actions.impl.ActionConfig`

Field	Type	Value	Description
repository	string	repository	The repository name.
workspace	string	collaboration	The workspace name.
action	ArrayList	<code>{java.util.ArrayList}</code>	The action name.

3.2.4. ContentTypeFilterPlugin

This plugin is used to filter WCM node types.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.templates.TemplateService</target-component>
```

The configuration is applied mainly in `/packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-templates-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.templates.TemplateService</target-component>
  <component-plugin>
    <name>FilterContentTypeForWCMSpecificFolder</name>
    <set-method>addContentTypeFilterPlugin</set-method>
    <type>org.exoplatform.services.cms.templates.ContentTypeFilterPlugin</type>
    <description>this plugin is used to filter wcm nodetype</description>
    <init-params>
      <object-param>
        <name>cssFolderFilter</name>
        <description>only exo:cssFile can be created in exo:cssFolder</description>
        <object type="org.exoplatform.services.cms.templates.ContentTypeFilterPlugin$FolderFilterConfig">
          <field name="folderType">
            <string>exo:cssFolder</string>
          </field>
          <field name="contentTypes">
            <collection type="java.util.ArrayList">
              <value>
                <string>exo:cssFile</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
      <object-param>
        ...
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name:** `FilterContentTypeForWCMSpecificFolder`
- **Set-method:** `addContentTypeFilterPlugin`

- **Type:** `org.exoplatform.services.cms.templates.ContentTypeFilterPlugin`
- **Object type:** `org.exoplatform.services.cms.templates.ContentTypeFilterPlugin$FolderFilterConfig`

Field	Type	Value	Description
folderType	string	<code>exo:cssFolder</code>	The folder type.
contentTypes	ArrayList	<code>{java.util.ArrayList}</code>	The content type.

3.2.5. ContextPlugin

This plugin is used to store the context configuration of a publication lifecycle. To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.wcm.extensions.publication.PublicationManager</target-component>
```

The configuration is applied mainly in `packaging/ecmdemo/webapp/src/main/webapp/WEB-INF/conf/sample-portal/wcm/content-configuration.xml` or `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/content-extended/authoring/configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.extensions.publication.PublicationManager</target-component>
  <component-plugin>
    <name>AddContext</name>
    <set-method>addContext</set-method>
    <type>org.exoplatform.services.wcm.extensions.publication.context.ContextPlugin</type>
    <init-params>
      <object-param>
        <name>contexts</name>
        <object type="org.exoplatform.services.wcm.extensions.publication.context.impl.ContextConfig">
          <field name="contexts">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.wcm.extensions.publication.context.impl.C">
                  <field name="name">
                    <string>context2</string>
                  </field>
                  <field name="priority">
                    <string>100</string>
                  </field>
                  <field name="lifecycle">
                    <string>lifecycle2</string>
                  </field>
                  <field name="site">
                    <string>acme</string>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name:** *AddContext*
- **Set-method:** *addContext*
- **Type:** *org.exoplatform.services.wcm.extensions.publication.context.ContextPlugin*
- **Object type:** *org.exoplatform.services.wcm.extensions.publication.context.impl.ContextConfig*

Field	Type	Value	Description
name	string	context2	The name of the context.
priority	string	100	The context priority, the higher number indicates higher priority. Because a site may have several lifecycles, the lifecycle with higher priority will be executed sooner.
lifecycle	string	lifecycle2	The name of the lifecycle.
site	string	acme	The site that will apply the context configuration.

3.2.6. CreatePortalPlugin

This is an abstract class and the parent of [axonomyPlugin](#) . You can create a link from this plugin to its children.

3.2.7. ExcludeIncludeDataTypePlugin

This plugin is used in the [SiteSearchService](#) component to filter the search results before these results are presented on the search page.

The configuration is applied mainly in *core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-search-configuration.xml*.

Sample configuration:

```
<component-plugins>
  <component-plugin>
    <name>ExcludeMimeTypes</name>
    <set-method>addExcludeIncludeDataTypePlugin</set-method>
    <type>org.exoplatform.services.wcm.search.ExcludeIncludeDataTypePlugin</type>
    <init-params>
      <properties-param>
        <name>search.exclude.datatypes</name>
        <description>exclude some data type when search</description>
        <property name="mimetypes" value="text/css,text/javascript,application/x-javascript,text/ecmascript"/>
      </properties-param>
    </init-params>
  </component-plugin>
</component-plugins>
```

In which:

- **Name:** ExcludeMimeTypes
- **Set-method:** addExcludeIncludeDataTypePlugin
- **Type:** org.exoplatform.services.wcm.search.ExcludeIncludeDataTypePlugin
- The plugin has the following parameter:

Properties-param	Description
search.exclude.datatype	Exclude some data types when doing search.

- The search.exclude.datatype property includes two attributes:

Attribute	Value	Description
name	mimetypes	The name of the property param.
value	text/css, text/javascript, application/javascript	The list of mimetypes which will be excluded from the search results.

3.2.8. FriendlyPlugin

This plugin is used to refine URLs in WCM.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.wcm.friendly.FriendlyService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/content-extended/friendly/configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.friendly.FriendlyService</target-component>
  <component-plugin>
    <name>FriendlyService.addConfiguration</name>
    <set-method>addConfiguration</set-method>
    <type>org.exoplatform.services.wcm.friendly.impl.FriendlyPlugin</type>
    <description>Configures</description>
    <priority>100</priority>
    <init-params>
      <value-param>
        <name>enabled</name>
        <value>true</value>
      </value-param>
      <object-param>
        <name>friendlies.configuration</name>
        <object type="org.exoplatform.services.wcm.friendly.impl.FriendlyConfig">
          <field name="friendlies">
            <collection type="java.util.ArrayList">
              <value>
```

```

        <object type="org.exoplatform.services.wcm.friendly.impl.FriendlyConfig$Friendly
          <field name="friendlyUri">
            <string>documents</string>
          </field>
          <field name="unfriendlyUri">
            <string>/public/acme/detail?content-id=/repository/collaboration</string>
          </field>
        </object>
      </value>
    <value>
      <object type="org.exoplatform.services.wcm.friendly.impl.FriendlyConfig$Friendly
        <field name="friendlyUri">
          <string>files</string>
        </field>
        <field name="unfriendlyUri">
          <string>/rest-ecmdemo/jcr/repository/collaboration</string>
        </field>
      </object>
    </value>
  </collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** *FriendlyService.addConfiguration*
- **Set-method:** *addConfiguration*
- **Type:** *org.exoplatform.services.wcm.friendly.impl.FriendlyPlugin*
- **Object type:** *org.exoplatform.services.wcm.friendly.impl.FriendlyConfig*

Field	Type	Value	Description
friendlyUrl	string	documents	The object that will be applied the friendly URL.
unfriendlyUrl	string	/public/acme/detail?content-id=/repository/collaboration	The path to the location that will be applied the friendly URL.

3.2.9. ImageThumbnailPlugin

This plugin is used to configure the file types and get thumbnail for images.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.thumbnail.ThumbnailService</target-component>
```

The configuration is applied mainly in *packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-thumbnail-configuration.xml*.

Sample configuration:

```

<external-component-plugins>
  <target-component>org.exoplatform.services.cms.thumbnail.ThumbnailService</target-component>
  <component-plugin>
    <name>ImageThumbnailPlugin</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.cms.thumbnail.impl.ImageThumbnailPlugin</type>
    <init-params>
      <object-param>
        <name>thumbnailType</name>
        <description>Thumbnail types</description>
        <object type="org.exoplatform.services.cms.thumbnail.impl.ThumbnailType">
          <field name="mimeTypes">
            <collection type="java.util.ArrayList">
              <value>
                <string>image/jpeg</string>
              </value>
              <value>
                <string>image/png</string>
              </value>
              <value>
                <string>image/gif</string>
              </value>
              <value>
                <string>image/bmp</string>
              </value>
              <value>
                <string>image/tiff</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

In which:

- **Name:** ImageThumbnailPlugin
- **Set-method:** addPlugin
- **Type:** org.exoplatform.services.cms.thumbnail.impl.ImageThumbnailPlugin
- **Object type:** org.exoplatform.services.cms.thumbnail.impl.ThumbnailType

Field	Type	Value	Description
mimeTypes	String	image/jpeg image/png image/gif image/bmp image/tiff	The list of thumbnail image types.

3.2.10. InitialWebcontentPlugin

When a portal is created, this plugin will deploy initial web-contents as the site artifact into the **Site Artifact** folder of that portal.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.wcm.portal.artifacts.CreatePortalArtifactsService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/wcm-extension/wcm/newsletter-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.portal.artifacts.CreatePortalArtifactsService</target-component>
  <component-plugin>
    <name>Initial webcontent artifact for each site</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.wcm.webcontent.InitialWebContentPlugin</type>
    <description>This plugin deploy some initial webcontent as site artifact to site artifact folder of portal
      a portal is
      created
    </description>
    <init-params>
      <object-param>
        <name>Portal logo data</name>
        <description>Deployment Descriptor</description>
        <object type="org.exoplatform.services.deployment.DeploymentDescriptor">
          <field name="target">
            <object type="org.exoplatform.services.deployment.DeploymentDescriptor$Target">
              <field name="repository">
                <string>repository</string>
              </field>
              <field name="workspace">
                <string>collaboration</string>
              </field>
              <field name="nodePath">
                <string>/sites content/live/{portalName}/web contents/site artifacts</string>
              </field>
            </object>
          </field>
          <field name="sourcePath">
            <string>war:/conf/sample-portal/wcm/artifacts/site-resources/acme-templates/Logo.xml</string>
          </field>
        </object>
      </object-param>
      <object-param>
        <name>Portal signin data</name>
        <description>Deployment Descriptor</description>
        <object type="org.exoplatform.services.deployment.DeploymentDescriptor">
          <field name="target">
            <object type="org.exoplatform.services.deployment.DeploymentDescriptor$Target">
              <field name="repository">
                <string>repository</string>
              </field>
              <field name="workspace">
                <string>collaboration</string>
              </field>
              <field name="nodePath">
                <string>/sites content/live/{portalName}/web contents/site artifacts</string>
              </field>
            </object>
          </field>
          <field name="sourcePath">
            <string>war:/conf/sample-portal/wcm/artifacts/site-resources/acme-templates/Signin.xml</string>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

    ...
  </object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** AddLifecycle
- **Set-method:** addPlugin
- **Type:** org.exoplatform.services.wcm.webcontent.InitialWebContentPlugin
- **Object type:** org.exoplatform.services.deployment.DeploymentDescriptor\$Target

Name	Type	Value	Description
repository	string	repository	The repository into which the initial web contents will be deployed.
workspace	string	collaboration	The workspace into which the initial web contents will be deployed.
nodePath	string	/sites content/live//web contents/site artifacts	The target node where the initial web-contents will be deployed into.
sourcePath	string	war:/conf/sample-portal/ Web/path to the sources	War path to the sources that this plugin will get data.

3.2.11. LinkDeploymentPlugin

This plugin is used to create predefined Symlinks into the system.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.deployment.WCMContentInitializerService</target-component>
```

The configuration is applied mainly in `packaging/ecmdemo/webapp/src/main/webapp/WEB-INF/conf/sample-portal/wcm/deployment/acme-deployment-configuration.xml`

Sample configuration:

```

<external-component-plugins>
  <target-component>org.exoplatform.services.deployment.WCMContentInitializerService</target-component>
  <component-plugin>
    <name>Content Initializer Service</name>
    <set-method>addPlugin</set-method>
  
```

```

<type>org.exoplatform.services.deployment.plugins.LinkDeploymentPlugin</type>
<description>Link Deployment Plugin</description>
<init-params>
  <object-param>
    <name>link01</name>
    <description>Deployment Descriptor</description>
    <object type="org.exoplatform.services.deployment.plugins.LinkDeploymentDescriptor">
      <field name="sourcePath">
        <string>repository:collaboration:/sites content/live/acme/web contents/News/News1</string>
      </field>
      <field name="targetPath">
        <string>repository:collaboration:/sites content/live/acme/categories/acme</string>
      </field>
    </object>
  </object-param>
  <object-param>
    <name>link02</name>
    <description>Deployment Descriptor</description>
    <object type="org.exoplatform.services.deployment.plugins.LinkDeploymentDescriptor">
      <field name="sourcePath">
        <string>repository:collaboration:/sites content/live/acme/web contents/News/News2</string>
      </field>
      <field name="targetPath">
        <string>repository:collaboration:/sites content/live/acme/categories/acme</string>
      </field>
    </object>
  </object-param>
  <object-param>
    <name>link03</name>
    <description>Deployment Descriptor</description>
    <object type="org.exoplatform.services.deployment.plugins.LinkDeploymentDescriptor">
      <field name="sourcePath">
        <string>repository:collaboration:/sites content/live/acme/web contents/News/News3</string>
      </field>
      <field name="targetPath">
        <string>repository:collaboration:/sites content/live/acme/categories/acme</string>
      </field>
    </object>
  </object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** *Content Initializer Service*
- **Set-method:** *addPlugin*
- **Type:** *org.exoplatform.services.wcm.webcontent.InitialWebContentPlugin*
- **Object type:** *org.exoplatform.services.deployment.plugins.LinkDeploymentDescriptor*

Field	Type	Value	Description
sourcePath	string	repository:collaboration:/sites content/live/acme/web contents/News/News1	The path to the source where this plugin will get data.
targetPath	string	repository:collaboration:/sites content/live/acme/categories/acme	The path to the target where this plugin will deploy.

3.2.12. LockGroupsOrUsersPlugin

This plugin is used to configure predefined groups or users for lock administration. To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.lock.LockService</target-component>
```

The configuration is applied mainly in `core/core-configuration/src/main/webapp/WEB-INF/conf/wcm-core/core-services-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.lock.LockService</target-component>
  <component-plugin>
    <name>predefinedLockGroupsOrUsersPlugin</name>
    <set-method>addLockGroupsOrUsersPlugin</set-method>
    <type>org.exoplatform.services.cms.lock.impl.LockGroupsOrUsersPlugin</type>
    <init-params>
      <object-param>
        <name>LockGroupsOrUsers.configuration</name>
        <description>configuration predefined groups or users for lock administrator</description>
        <object type="org.exoplatform.services.cms.lock.impl.LockGroupsOrUsersConfig">
          <field name="settingLockList">
            <collection type="java.util.ArrayList">
              <value>
                <string>*:/platform/administrators</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name:** `predefinedLockGroupsOrUsersPlugin`
- **Set-method:** `addLockGroupsOrUsersPlugin`
- **Type:** `org.exoplatform.services.cms.lock.impl.LockGroupsOrUsersPlugin`
- **Object type:** `org.exoplatform.services.cms.lock.impl.LockGroupsOrUsersConfig`

Field	Type	Value	Description
settingLockList	ArrayList	<code>{java.util.ArrayList}</code>	The list of the groups or user to be locked.

3.2.13. ManageDrivePlugin

This plugin is used to create a predefined drive into a repository. A drive can be considered as a shortcut in the content repository, a quick access to some places for users. You can restrict the visibility of this drive to a group/user and apply a specific view depending on the content you have in this area.

A drive is the combination of:

- Path: the root folder of the drive.
- View: how we can see contents, such as by list, thumbnails, coverflow.
- Role: the visibility to every users, a group or a single user.
- Options: allow you to specify whether to see hidden nodes or not and to create folders in this drive or not.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.drives.ManageDriveService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-drives-configuration.xml`.

The following structure is used for drives configuration.

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.drives.ManageDriveService</target-component>
  <component-plugin>
    <name>manage.drive.plugin</name>
    <set-method>setManageDrivePlugin</set-method>
    <type>org.exoplatform.services.cms.drives.impl.ManageDrivePlugin</type>
    <description>Nothing</description>
    <init-params>
      <object-param>
        There are initializing attributes of org.exoplatform.services.cms.drives.DriveData object
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

The file that contains the structure above will be configured in the `configuration.xml` file as the following:

```
<import>war:/conf/wcm-extension/dms/drives-configuration.xml</import>
```

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.drives.ManageDriveService</target-component>
  <component-plugin>
    <name>manage.drive.plugin</name>
    <set-method>setManageDrivePlugin</set-method>
    <type>org.exoplatform.services.cms.drives.impl.ManageDrivePlugin</type>
    <description>Nothing</description>
    <init-params>
      <object-param>
        <name>Managed Sites</name>
        <description>Managed Sites</description>
        <object type="org.exoplatform.services.cms.drives.DriveData">
          <field name="name">
            <string>Managed Sites</string>
          </field>
          <field name="repository">
            <string>repository</string>
          </field>
          <field name="workspace">
            <string>collaboration</string>
          </field>
          <field name="permissions">
            <string>*:/platform/administrators</string>
          </field>
          <field name="homePath">

```

```

        <string>/sites content/live</string>
      </field>
      <field name="icon">
        <string/>
      </field>
      <field name="views">
        <string>wcm-view</string>
      </field>
      <field name="viewPreferences">
        <boolean>>false</boolean>
      </field>
      <field name="viewNonDocument">
        <boolean>>true</boolean>
      </field>
      <field name="viewSideBar">
        <boolean>>true</boolean>
      </field>
      <field name="showHiddenNode">
        <boolean>>false</boolean>
      </field>
      <field name="allowCreateFolders">
        <string>nt: folder, nt: unstructured</string>
      </field>
      <field name="allowNodeTypesOnTree">
        <string>*</string>
      </field>
    </object>
  </object-param>
  <object-param>
    <name>Public</name>
    <description>Public drive</description>
    <object type="org.exoplatform.services.cms.drives.DriveData">
      <field name="name">
        <string>Public</string>
      </field>
      <field name="repository">
        <string>repository</string>
      </field>
      <field name="workspace">
        <string>collaboration</string>
      </field>
      <field name="permissions">
        <string>*/platform/users</string>
      </field>
      <field name="homePath">
        <string>/Users/${userId}/Public</string>
      </field>
      <field name="icon">
        <string/>
      </field>
      <field name="views">
        <string>simple-view, admin-view</string>
      </field>
      <field name="viewPreferences">
        <boolean>>false</boolean>
      </field>
      <field name="viewNonDocument">
        <boolean>>false</boolean>
      </field>
      <field name="viewSideBar">
        <boolean>>true</boolean>
      </field>
      <field name="showHiddenNode">
        <boolean>>false</boolean>
      </field>
      <field name="allowCreateFolders">
        <string>nt: folder, nt: unstructured</string>
      </field>
      <field name="allowNodeTypesOnTree">
        <string>*</string>
      </field>
    </object>
  </object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** `manage.drive.plugin`
- **Set-method:** `setManageDrivePlugin`
- **Type:** `org.exoplatform.services.cms.drives.impl.ManageDrivePlugin`
- **Object type:** `org.exoplatform.services.cms.drives.DriveData`

Field	Type	Value	Description
name	String	Public	The name of drive which must be unique.
repository	String	repository	Content Repository where to find the root path.
workspace	String	collaboration	Workspace in the Content Repository.
homePath	String	/sites content/live	Root path in the Content Repository. userId can be used to use the <code>userId</code> at runtime in the path.
permissions	String	<i>*/platform/administrators</i>	Visibility of the drive based on eXo rights. For example: <i>*/platform/users</i>
icon	String	N/A	The Url to the icon.
views	String	wcm-view	The list of views you want to use, separated by commas. For example: <code>simple-view,admin-view</code>
viewPreferences	Boolean	false	The User Preference icon will be visible if true.
viewNonDocument	Boolean	true	Non-document types will be visible in the user view if true.
viewSideBar	Boolean	true	Show/Hide the left bar (with navigation and filters).
showHiddenNode	Boolean	false	Hidden nodes will be visible if true.
allowCreateFolders	String	nt:folder,nt:unstructured	List of node types that you can create as folders. For example:

Field	Type	Value	Description
			nt:folder,nt:unstructured.
allowNodeTypesOnTree	String	*	Allow you to filter node types in the navigation tree. For example, the default value is "*" to show all content types.

3.2.14. ManageViewPlugin

This plugin is used to create a predefined View into a repository. A View can include many object parameters. Parameters are used to create default Views, Templates and Actions of **Manage View** service. View enables administrators to customize View classification that can impact on users in exploring workspace. Each object-param has a type that is a class representing all properties of a View.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.views.ManageViewService</target-component>
```

The configuration is applied mainly in *packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-views-configuration.xml*.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.views.ManageViewService</target-component>
  <component-plugin>
    <name>manage.view.plugin</name>
    <set-method>setManageViewPlugin</set-method>
    <type>org.exoplatform.services.cms.views.impl.ManageViewPlugin</type>
    <description>this plugin manage user view</description>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>predefinedViewsLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>System-View</name>
        <description>View configuration for System workspace</description>
        <object type="org.exoplatform.services.cms.views.ViewConfig">
          <field name="name">
            <string>system-view</string>
          </field>
          <field name="permissions">
            <string>*:/platform/administrators</string>
          </field>
          <field name="template">
            <string>/exo:ecm/views/templates/ecm-explorer/SystemView</string>
          </field>
          <field name="tabList">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.views.ViewConfig$Tab">
```

```

        <field name="tabName">
            <string>Info</string>
        </field>
        <field name="buttons">
            <string>viewNodeType; viewPermissions; viewProperties; showJCRStructures</string>
        </field>
    </object>
</value>
</collection>
</field>
</object>
</object-param>
<object-param>
    <name>System Template</name>
    <description>Template for display documents in list style</description>
    <object type="org.exoplatform.services.cms.views.TemplateConfig">
        <field name="type">
            <string>ecmExplorerTemplate</string>
        </field>
        <field name="name">
            <string>SystemView</string>
        </field>
        <field name="warPath">
            <string>/ecm-explorer/SystemView.gtmpl</string>
        </field>
    </object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** *manage.view.plugin*
- **Set-method:** *setManageViewPlugin*
- **Type:** *org.exoplatform.services.cms.views.impl.ManageViewPlugin*
- **Init-param:**

value-param	Type	Value	Description
autoCreateInNewRepository	boolean	true	Allow creating a predefined View in this repository if the value is "true".
predefinedViewsLocation	string	war:/conf/dms-extension/dms-templates	The location of the View node in the repository.
repository	string	repository	The repository name.

- **Object type:** *org.exoplatform.services.cms.views.ViewConfig*

Field	Type	Value	Description
name	string	system-view	The name of view which must be unique inside WCM.

Field	Type	Value	Description
permissions	string	<code>*:/platform/administrators</code>	Visibility of the view based on eXo rights.
template	string	<code>/exo:ecm/views/templates</code>	Specify the path to the template location.
tabList	ArrayList	<code>{java.util.ArrayList}</code>	Include a set of view names.

- **Object type:** *org.exoplatform.services.cms.views.ViewConfig\$Tab*

Field	Type	Value	Description
tabName	string	Info	The name of tab which must be unique.
button	string	viewNodeType; viewPermissions; viewProperties; showJCRStructure	Specify a set of view component names.

- **Object type:** *org.exoplatform.services.cms.views.TemplateConfig*

Field	Type	Vsalue	Description
type	string	ecmExplorerTemplate	Specify if a name is truly a class representing all properties of a view.
name	string	system-view	Specify a set of view component names.
warPath	string	<code>/ecm-explorer/SystemView</code>	Specify a template location to view.

3.2.15. PDFThumbnailPlugin

This plugin is to set the supported file types of PDF thumbnail. See also [ImageThumbnailPlugin](#) .

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.thumbnail.ThumbnailService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-thumbnail-configuration.xml`.

Sample configuration:

```

<component-plugin>
  <name>PDFThumbnailPlugin</name>
  <set-method>addPlugin</set-method>
  <type>org.exoplatform.services.cms.thumbnail.impl.PDFThumbnailPlugin</type>
  <init-params>
    <object-param>
      <name>thumbnailType</name>
      <description>Thumbnail types</description>
      <object type="org.exoplatform.services.cms.thumbnail.impl.ThumbnailType">
        <field name="mimeTypes">
          <collection type="java.util.ArrayList">
            <value>
              <string>application/pdf</string>
            </value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component-plugin>

```

In which:

- **Name:** *PDFThumbnailPlugin*
- **Set-method:** *addPlugin*
- **Type:** *org.exoplatform.services.cms.thumbnail.impl.PDFThumbnailPlugin*
- **Object type:** *org.exoplatform.services.cms.thumbnail.impl.ThumbnailType*

Field	Type	Value	Description
mimeTypes	String	application/pdf	The MIME type of the pdf thumbnail.

3.2.16. PorletTemplatePlugin

This plugin is used to import the view templates into Content List Viewer.

To use the plugin in the component configuration, you must use the following target-component:

```

<target-component>org.exoplatform.services.cms.views.ApplicationTemplateManagerService</target-component>

```

The configuration is applied mainly in *packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/wcm-extension/dms/application-templates-configuration.xml*.

Sample configuration:

```

<external-component-plugins>
  <target-component>org.exoplatform.services.cms.views.ApplicationTemplateManagerService</target-component>
  <component-plugin>
    <name>clv.templates.plugin</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.cms.views.PortletTemplatePlugin</type>
    <description>This plugin is used to import views templates for Content List Viewer</description>
    <init-params>
      <value-param>
        <name>portletName</name>
      </value-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

```

        <value>content-list-viewer</value>
      </value-param>
      <value-param>
        <name>portlet.template.path</name>
        <value>war:/conf/wcm-artifacts/application-templates/content-list-viewer</value>
      </value-param>
      <object-param>
        <name>default.folder.list.viewer</name>
        <description>Default folder list viewer groovy template</description>
        <object type="org.exoplatform.services.cms.views.PortletTemplatePlugin$PortletTemplateConfig">
          <field name="templateName">
            <string>UIContentListPresentationDefault.gtmpl</string>
          </field>
          <field name="category">
            <string>list</string>
          </field>
        </object>
      </object-param>
      <object-param>
        ....
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

In which:

- **Name:** *clv.templates.plugin*
- **Set-method:** *addPlugin*
- **Type:** *org.exoplatform.services.cms.views.PortletTemplatePlugin*
- **Init-param:**

Value-param	Type	Value	Description
portletName	string	content-list-viewer	The name of the portlet.
portlet.template.path	string	war:/conf/wcm-artifacts/application-templates/content-list-viewer	The path to the configuration of the portlet.

- **Object type:** *org.exoplatform.services.cms.views.PortletTemplatePlugin\$PortletTemplateConfig*

Field	Type	Description
templateName	string	The name of the GROOVY template.
category	string	The category name.

3.2.17. PredefinedProcessesPlugin

This plugin is used to import the predefined processes into the system.

To use the plugin in the component configuration, you must use the following target-component:

```

<!-- Target component configuration placeholder -->

```

```
<target-component>org.exoplatform.services.workflow.WorkflowServiceContainer</target-component>
```

The configuration is applied mainly in `packaging/workflow/webapp/src/main/webapp/WEB-INF/workflow-extension/workflow/bonita-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.workflow.WorkflowServiceContainer</target-component>
  <component-plugin>
    <name>deploy.predefined.processes</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.workflow.PredefinedProcessesPlugin</type>
    <init-params>
      <object-param>
        <name>predefined.processes</name>
        <description>load of default business processes</description>
        <object type="org.exoplatform.services.workflow.ProcessesConfig">
          <field name="processLocation">
            <string>war:/conf/bp</string>
          </field>
          <field name="predefinedProcess">
            <collection type="java.util.HashSet">
              <value>
                <string>/exo-ecms-ext-workflow-bp-bonita-content-2.3.3.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-bonita-payraise-2.3.3.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-bonita-holiday-2.3.3.jar</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **name:** `deploy.predefined.processes`
- **set-method:** `addPlugin`
- **type:** `org.exoplatform.services.workflow.PredefinedProcessesPlugin`
- **Object type:** `org.exoplatform.services.workflow.ProcessesConfig`

Field	Type	Value	Description
processLocation	string	war:/conf/bp	The path to the process.
predefinedProcess	HashSet	java.util.HashSet	The list of the processes.

3.2.18. PublicationPlugin

This is an abstract class and the parent of [StageAndVersionPublicationPlugin](#) and [AuthoringPublicationPlugin](#). You can create a link from this plugin to its children.

3.2.19. QueryPlugin

This plugin is used to store predefined queries into the repositories of the system.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.queries.QueryService</target-component>
```

The configuration is applied mainly in `/packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-queries-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.queries.QueryService</target-component>
  <component-plugin>
    <name>query.plugin</name>
    <set-method>setQueryPlugin</set-method>
    <type>org.exoplatform.services.cms.queries.impl.QueryPlugin</type>
    <description>Nothing</description>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>>true</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>CreatedDocuments</name>
        <description>documents created by the current user</description>
        <object type="org.exoplatform.services.cms.queries.impl.QueryData">
          <field name="name">
            <string>Created Documents</string>
          </field>
          <field name="language">
            <string>xpath</string>
          </field>
          <field name="statement">
            <string>//*[(@jcr:primaryType = 'exo:article' or @jcr:primaryType = 'nt:file') and
              @exo:owner='${UserId}$'] order by @exo:dateCreated descending
            </string>
          </field>
          <field name="permissions">
            <collection type="java.util.ArrayList">
              <value>
                <string>*:/platform/users</string>
              </value>
            </collection>
          </field>
          <field name="cachedResult">
            <boolean>>false</boolean>
          </field>
        </object>
      </object-param>
      <object-param>
        <name>CreatedDocumentsDayBefore</name>
        <description>documents created the day before</description>
        <object type="org.exoplatform.services.cms.queries.impl.QueryData">
          <field name="name">
            <string>Created Document DayBefore</string>
          </field>
          <field name="language">
            <string>xpath</string>
          </field>
          <field name="statement">
            <string>//element(*,exo:article)[@exo:dateCreated < xs:dateTime('${Date}$')] order by
              @exo:dateCreated descending
            </string>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        <field name="permissions">
            <collection type="java.util.ArrayList">
                <value>
                    <string>*:/platform/users</string>
                </value>
            </collection>
        </field>
        <field name="cachedResult">
            <boolean>true</boolean>
        </field>
    </object>
</object-param>
<object-param>
    <name>AllArticles</name>
    <description>All articles</description>
    <object type="org.exoplatform.services.cms.queries.impl.QueryData">
        <field name="name">
            <string>All Articles</string>
        </field>
        <field name="language">
            <string>xpath</string>
        </field>
        <field name="statement">
            <string>//element(*,exo:article) order by @exo:dateCreated descending</string>
        </field>
        <field name="permissions">
            <collection type="java.util.ArrayList">
                <value>
                    <string>*:/platform/users</string>
                </value>
            </collection>
        </field>
        <field name="cachedResult">
            <boolean>true</boolean>
        </field>
    </object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** predefinedTaxonomyPlugin
- **Set-method:** setQueryPlugin
- **Type:** org.exoplatform.services.cms.queries.impl.QueryPlugin
- **Init-param:**

Value-param	Type	Value	Description
autoCreateInNewRepository	boolean	true	Store queries in a new repository if the value is "true".
repository	string	repository	The repository to the target node.

- **Object type:** org.exoplatform.services.cms.queries.impl.QueryData

Field	Type	Description
name	string	The name of the query.
language	string	The language of the query (Xpath, SQL).
statement	string	The query statement.
permissions	ArrayList	The permission which users must have to use this query.
cachedResult	boolean	Specify if the query is cached or not.

3.2.20. RemovePortalPlugin

This is an abstract class and the parent of [RemoveTaxonomyPlugin](#) . You can create a link from this plugin to its children.

3.2.21. RemoveTaxonomyPlugin

This plugin is used to invalidate taxonomy trees in **categories** folder of a portal when the portal is removed.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.wcm.portal.artifacts.RemovePortalArtifactsService</target-component>
```

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.portal.artifacts.RemovePortalArtifactsService</target-component>
  <component-plugin>
    <name>Remove taxonomy tree</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.wcm.category.RemoveTaxonomyPlugin</type>
    <description>This plugin invalidate taxonomy tree to categories folder of portal when a portal is removed</description>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name:** Remove taxonomy tree
- **Set-method:** addPlugin
- **Type:** org.exoplatform.services.wcm.category.RemoveTaxonomyPlugin

3.2.22. ScriptActionPlugin

This plugin is used to import the predefined script actions into the system.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.actions.ActionServiceContainer</target-component>
```

The configuration is applied mainly in *packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-actions-configuration.xml*.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.actions.ActionServiceContainer</target-component>
  <component-plugin>
    <name>exo:scriptAction</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.cms.actions.impl.ScriptActionPlugin</type>
    <init-params>
      <object-param>
        <name>predefined.actions</name>
        <description>description</description>
        <object type="org.exoplatform.services.cms.actions.impl.ActionConfig">
          <field name="repository">
            <string>repository</string>
          </field>
          <field name="workspace">
            <string>collaboration</string>
          </field>
          <field name="actions">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Action">
                  <field name="type">
                    <string>exo:sendMailAction</string>
                  </field>
                  <field name="name">
                    <string>sendMail</string>
                  </field>
                  <field name="description">
                    <string>send a notification mail</string>
                  </field>
                  <field name="srcWorkspace">
                    <string>collaboration</string>
                  </field>
                  <field name="srcPath">
                    <string>/Documents/Validation Requests</string>
                  </field>
                  <field name="isDeep">
                    <boolean>true</boolean>
                  </field>
                  <field name="lifecyclePhase">
                    <collection type="java.util.ArrayList">
                      <value>
                        <string>read</string>
                      </value>
                    </collection>
                  </field>
                  <field name="roles">
                    <string>*:/platform/administrators</string>
                  </field>
                  <field name="variables">
                    <string>exo:to=benjamin.mestrallet@exoplatform.com</string>
                  </field>
                  <field name="mixins">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.actions.impl.ActionCo
                          <field name="name">
                            <string>mix:affectedNodeTypes</string>
                          </field>
                          <field name="properties">
                            <string>
                              exo:affectedNodeTypeNames=exo:article,exo:podcast,ex
                            </string>
                          </field>
                        </object>
                      </value>
                    </collection>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        </field>
      </object>
    </value>
  </collection>
</field>
</object>
</value>
<value>
  <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Action">
    <field name="type">
      <string>exo:trashFolderAction</string>
    </field>
    <field name="name">
      <string>trashFolder</string>
    </field>
    <field name="description">
      <string>trigger actions for items in trash</string>
    </field>
    <field name="srcWorkspace">
      <string>collaboration</string>
    </field>
    <field name="srcPath">
      <string>/Trash</string>
    </field>
    <field name="isDeep">
      <boolean>false</boolean>
    </field>
    <field name="lifecyclePhase">
      <collection type="java.util.ArrayList">
        <value>
          <string>node_added</string>
        </value>
        <value>
          <string>node_removed</string>
        </value>
      </collection>
    </field>
  </object>
</value>
</collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **name:** *exo:scriptAction_*
- **set-method :** *addPlugin*
- **type:** *org.exoplatform.services.cms.actions.impl.ScriptActionPlugin*
- **Object type:** *org.exoplatform.services.cms.actions.impl.ActionConfig*

Name	Type	Default Value	Description
repository	string	repository	The name of the repository.
workspace	string	collaboration	The name of the workspace.
actions	ArrayList	{ <i>java.util.ArrayList</i> }	The list of the actions.

- **Object type:** `org.exoplatform.services.cms.actions.impl.ActionConfig$Action`

Name	Type	Default Value	Description
type	string	<code>exo:sendMailAction</code>	The type of the action.
name	string	<code>sendMail</code>	The name of the action.
description	string	<code>send a notification mail</code>	The description of the action.
srcWorkspace	string	<code>collaboration</code>	The source workspace of the action.
isDeep	boolean	<code>false</code>	Specify the depth of node that the action script will affect.
srcPath	string	<code>trash</code>	The path to the source.
lifecyclePhase	ArrayList	<code>{java.util.ArrayList}</code>	Specify the lifecycle phase that the action will take place.

3.2.23. ScriptPlugin

This plugin is used to add groovy scripts into the system.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.scripts.ScriptService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-scripts-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.scripts.ScriptService</target-component>
  <component-plugin>
    <name>manage.script.plugin</name>
    <set-method>addScriptPlugin</set-method>
    <type>org.exoplatform.services.cms.scripts.impl.ScriptPlugin</type>
    <description>Nothing</description>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <value-param>
        <name>predefinedScriptsLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts</value>
      </value-param>
      <object-param>
        <name>predefined.scripts</name>
        <description>description</description>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

<object type="org.exoplatform.services.cms.impl.ResourceConfig">
  <field name="resources">
    <collection type="java.util.ArrayList">
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/RSSScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/SendMailScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/TrashFolderScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/EnableVersioningScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/AutoVersioningScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/AddMetadataScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/TransformBinaryChildrenToTextScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/GetMailScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/ProcessRecordsScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/PublishingRequestScript.groovy</string>
          </field>
        </object>
      </value>
      <value>
        <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
          <field name="name">
            <string>ecm-explorer/action/AddTaxonomyActionScript.groovy</string>
          </field>
        </object>
      </value>
    </collection>
  </field>
</object>

```

```

        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/widget/FillSelectBoxWithCalendarCategories.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/widget/FillSelectBoxWithMetadatas.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/widget/FillSelectBoxWithWorkspaces.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/widget/FillSelectBoxWithNodeChildren.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/widget/FillSelectBoxWithLanguage.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/interceptor/PreNodeSaveInterceptor.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/interceptor/PostNodeSaveInterceptor.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>ecm-explorer/interceptor/PostFilePlanInterceptor.groovy</string>
                </field>
            </object>
        </value>
        <value>
            <object type="org.exoplatform.services.cms.impl.ResourceConfig$Resource">
                <field name="name">
                    <string>content-browser/GetDocuments.groovy</string>
                </field>
            </object>
        </value>
    </collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** manage.script.plugin

- **Set-method:** addScriptPlugin
- **Type:** org.exoplatform.services.cms.scripts.impl.ScriptPlugin
- **Init-param:**

Value-param	Type	Value	Description
autoCreateInNewRepository	boolean	true	Enable/Disable the creation of the scripts in the newly created repository.
repository	String	repository	The repository name.
predefinedScriptsLocation	String	war:/conf/dms-extension/dms-location	The location where the scripts are created.

- **Object type:** org.exoplatform.services.cms.impl.ResourceConfig

Field	Type	Value	Description
resource	ArrayList	{java.util.ArrayList}	The resource name.

3.2.24. StageAndVersionPublicationPlugin

This plugin is used to control the state life cycle of a content.

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/content-extended/authoring/configuration.xml`.

Sample configuration:

```
<component-plugin>
  <name>States and versions based publication</name>
  <set-method>addPublicationPlugin</set-method>
  <type>org.exoplatform.services.wcm.publication.lifecycle.stageversion.StageAndVersionPublicationPlugin</type>
  <description>This publication lifecycle publish a web content or DMS document to a portal page with more states
    and version.
  </description>
</component-plugin>
```

In which:

- **name:** *States and versions based publication*
- **set method:** *addPublicationPlugin*
- **type:** *org.exoplatform.services.wcm.publication.lifecycle.stageversion.StageAndVersionPublicationPlugin*

3.2.25. StatesLifecyclePlugin

This plugin is used to control the state life cycle of a content.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.wcm.extensions.publication.PublicationManager</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/content-extended/authoring/configuration.xml`.

Sample configuration:

```
<component-plugin>
  <name>AddLifecycle</name>
  <set-method>addLifecycle</set-method>
  <type>org.exoplatform.services.wcm.extensions.publication.lifecycle.StatesLifecyclePlugin</type>
  <description>Configures</description>
  <priority>1</priority>
  <init-params>
    <object-param>
      <name>lifecycles</name>
      <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig">
        <field name="lifecycles">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.Li
                <field name="name">
                  <string>lifecycle1</string>
                </field>
                <field name="publicationPlugin">
                  <string>Authoring publication</string>
                </field>
                <field name="states">
                  <collection type="java.util.ArrayList">
                    <value>
                      <object type="org.exoplatform.services.wcm.extensions.publication.li
                        <field name="state">
                          <string>draft</string>
                        </field>
                        <field name="membership">
                          <string>author:/platform/web-contributors</string>
                        </field>
                      </object>
                    </value>
                    <value>
                      <object type="org.exoplatform.services.wcm.extensions.publication.li
                        <field name="state">
                          <string>pending</string>
                        </field>
                        <field name="membership">
                          <string>author:/platform/web-contributors</string>
                        </field>
                      </object>
                    </value>
                    <value>
                      <object type="org.exoplatform.services.wcm.extensions.publication.li
                        <field name="state">
                          <string>approved</string>
                        </field>
                        <field name="membership">
                          <string>manager:/platform/web-contributors</string>
                        </field>
                      </object>
                    </value>
                    <value>
                      <object type="org.exoplatform.services.wcm.extensions.publication.li
                        <field name="state">
                          <string>staged</string>
                        </field>
                        <field name="membership">
                          <string>publisher:/platform/web-contributors</string>
                        </field>
                      </object>
                    </value>
                  </collection>
                </field>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
```

```

</value>
<value>
  <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig$Lifecycle"
    <field name="state">
      <string>published</string>
    </field>
    <field name="membership">
      <string>publisher:/platform/web-contributors</string>
    </field>
  </object>
</value>
</collection>
</field>
</object>
</value>
</collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>

```

In which:

- **Name:** *AddLifecycle*
- **Set-method:** *addLifecycle*
- **Type:** *org.exoplatform.services.wcm.extensions.publication.lifecycle.StatesLifecyclePlugin*
- **Object** **type:**
org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig\$Lifecycle

Field	Type	Value	Description
name	string	lifecycle1	The name of the lifecycle.
publicationPlugin	string	Authoring publication	The publication plugin name.
states	ArrayList	<i>{java.util.ArrayList}</i>	The list of the publication states.

- **Object** **type:**
org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig\$State

Field	Type	Description
state	string	The publication states: draft, pending, staged, approved or published.
membership	string	The user or group.

3.2.26. TagPermissionPlugin

This plugin is used to configure the predefined permission for tag to inject in JCR.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.folksonomy.NewFolksonomyService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-folksonomy-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.folksonomy.NewFolksonomyService</target-component>
  <component-plugin>
    <name>predefinedTagPermissionPlugin</name>
    <set-method>addTagPermissionPlugin</set-method>
    <type>org.exoplatform.services.cms.folksonomy.impl.TagPermissionPlugin</type>
    <init-params>
      <object-param>
        <name>TagPermission.configuration</name>
        <description>configuration predefined permission for tag to inject in jcr</description>
        <object type="org.exoplatform.services.cms.folksonomy.impl.TagPermissionConfig">
          <field name="tagPermissionList">
            <collection type="java.util.ArrayList">
              <value>
                <string>*:/platform/administrators</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name:** `predefinedTagPermissionPlugin`
- **Set-method:** `addTagPermissionPlugin`
- **Type:** `org.exoplatform.services.cms.folksonomy.impl.TagPermissionPlugin`
- **Object type:** `org.exoplatform.services.cms.folksonomy.impl.TagPermissionConfig`

Name	Type	Value	Description
tagPermissionList	ArrayList	{java.util.ArrayList}	The users/groups that have the permission.

3.2.27. TagStylePlugin

This plugin is used to configure the predefined styles for tag to inject in JCR.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.folksonomy.NewFolksonomyService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-folksonomy-configuration.xml`.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.folksonomy.NewFolksonomyService</target-component>
  <component-plugin>
    <name>predefinedTagStylePlugin</name>
    <set-method>addTagStylePlugin</set-method>
    <type>org.exoplatform.services.cms.folksonomy.impl.TagStylePlugin</type>
    <init-params>
      <object-param>
        <name>htmStyleForTag.configuration</name>
        <description>configuration predefined html style for tag to inject in jcr</description>
        <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig">
          <field name="autoCreatedInNewRepository">
            <boolean>true</boolean>
          </field>
          <field name="repository">
            <string>repository</string>
          </field>
          <field name="tagStyleList">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTagStyle">
                  <field name="name">
                    <string>normal</string>
                  </field>
                  <field name="tagRate">
                    <string>0..2</string>
                  </field>
                  <field name="htmlStyle">
                    <string>font-size: 12px; font-weight: bold; color: #6b6b6b; font-family:
                      verdana; text-decoration:none;
                  </string>
                  </field>
                  <field name="description">
                    <string>Normal style for tag</string>
                  </field>
                </object>
              </value>
              <value>
                <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTagStyle">
                  <field name="name">
                    <string>interesting</string>
                  </field>
                  <field name="tagRate">
                    <string>2..5</string>
                  </field>
                  <field name="htmlStyle">
                    <string>font-size: 13px; font-weight: bold; color: #5a66ce; font-family:
                      verdana; text-decoration:none;
                  </string>
                  </field>
                  <field name="description">
                    <string>Interesting style for tag</string>
                  </field>
                </object>
              </value>
              <value>
                <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTagStyle">
                  <field name="name">
                    <string>attractive</string>
                  </field>
                  <field name="tagRate">
                    <string>5..7</string>
                  </field>
                  <field name="htmlStyle">
                    <string>font-size: 15px; font-weight: bold; color: blue; font-family: Ar
                      text-decoration:none;
                  </string>
                  </field>
                  <field name="description">
                    <string>attractive style for tag</string>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </init-params>
    </component-plugin>
  </external-component-plugins>
```

```

        </field>
      </object>
    </value>
    <value>
      <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTa
        <field name="name">
          <string>hot</string>
        </field>
        <field name="tagRate">
          <string>7..10</string>
        </field>
        <field name="htmlStyle">
          <string>font-size: 18px; font-weight: bold; color: #ff9000; font-family:
            text-decoration:none;
          </string>
        </field>
        <field name="description">
          <string>hot style for tag</string>
        </field>
      </object>
    </value>
    <value>
      <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTa
        <field name="name">
          <string>hottest</string>
        </field>
        <field name="tagRate">
          <string>10..*</string>
        </field>
        <field name="htmlStyle">
          <string>font-size: 20px; font-weight: bold; color: red; font-family:Aria
            text-decoration:none;
          </string>
        </field>
        <field name="description">
          <string>hottest style for tag</string>
        </field>
      </object>
    </value>
  </collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

- **Name:** *predefinedTagStylePlugin*
- **Set-method:** *addTagStylePlugin*
- **Type:** *org.exoplatform.services.cms.folksonomy.impl.TagStylePlugin*
- **Object type:** *org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig*

Name	Type	Value	Description
autoCreatedInNewRepository	boolean	true	Specify whether the tag style is added automatically in a new repository or not.
repository	string	repository	Name of the repository where the tag style is added.
tagStyleList	ArrayList	{java.util.ArrayList}	The list of tag styles.

- **Object type:** *org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig\$HtmlTagStyle*

Name	Type	Description
name	string	The name of the tag.
tagRate	string	The number of times that a tag is used which will decide the respective tag style.
htmlStyle	string	The HTML code that defines the style.

3.2.28. TaxonomyPlugin

This plugin is used to configure the predefined taxonomies to inject into JCR.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.taxonomy.TaxonomyService</target-component>
```

The configuration is applied mainly in *packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-categories-configuration.xml*.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.taxonomy.TaxonomyService</target-component>
  <component-plugin>
    <name>predefinedTaxonomyPlugin</name>
    <set-method>addTaxonomyPlugin</set-method>
    <type>org.exoplatform.services.cms.taxonomy.impl.TaxonomyPlugin</type>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <value-param>
        <name>workspace</name>
        <value>dms-system</value>
      </value-param>
      <value-param>
        <name>treeName</name>
        <value>System</value>
      </value-param>
      <object-param>
        <name>permission.configuration</name>
        <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig">
          <field name="taxonomies">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Taxonomy">
                  <field name="permissions">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.taxonomy.impl.Taxonomy">
                          <field name="identity">
                            <string>*:/platform/users</string>
                          </field>
                        </object>
                      </value>
                    </collection>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

<field name="read">
  <string>true</string>
</field>
<field name="addNode">
  <string>true</string>
</field>
<field name="setProperty">
  <string>true</string>
</field>
<field name="remove">
  <string>false</string>
</field>
</object>
</value>
</collection>
</field>
</object>
</value>
</collection>
</field>
</object>
</object-param>
<object-param>
  ...
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **Name:** *predefinedTaxonomyPlugin*
- **Set-method:** *addTaxonomyPlugin*
- **Type:** *org.exoplatform.services.cms.taxonomy.impl.TaxonomyPlugin*
- **Init-param:**

Value-param	Type	Value	Description
autoCreateInNewRepository	boolean	true	Enable/Disable the creation of the taxonomies in the newly created repository.
repository	string	repository	The name of the repository where taxonomies are created.
workspace	string	dms-system	The name of the workspace where taxonomies are created.
treeName	string	system	The name of the taxonomy tree created.

- **Object type:** *org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig*

Name	Type	Value	Description
taxonomies	ArrayList	{java.util.ArrayList}	The list of taxonomies to be configured with permission.

- **Object type:** *org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig\$Taxonomy*

Name	Type	Value	Description
permissions	ArrayList	{java.util.ArrayList}	The list of permissions for users or groups to access the taxonomy.

- **Object type:** *org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig\$Permission*

Name	Type	Value	Description
identity	string	*:/platform/users	The name of the user, group or membership.
read	boolean	true	The permission to read the taxonomy tree.
addNode	boolean	true	The permission to add a node to the taxonomy tree.
setProperty	boolean	true	The permission to set properties for a node in the taxonomy tree.
remove	boolean	false	The permission to remove a node from the taxonomy tree.

3.2.29. TemplatePlugin

This plugin is used to create templates into the system. A template is a presentation to display the saved information.

The node type template is used to edit and display the node content. Each node type has one *dialog1.gtmpl* file (dialog template) for editing/creating a node and one *view1.gtmpl* file (view template) for viewing the node content. Using the dialog template, you can specify a dialog whose fields correspond to the properties of the node you want to edit their values. When this template is rendered, each specified field will appear with a data input box for you to edit. Note that you do not have to design a dialog in which all data of the node are listed to be edited. You can just list the subset of node data you want to edit. Like the dialog template, the view template renders information of the node. You just need to create the template and specify which data fields to be displayed. With this kind of template, node information is only displayed but cannot be edited. See details at [ContentType](#).

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.cms.templates.TemplateService</target-component>
```

The configuration is applied mainly in `packaging/wcm/webapp/src/main/webapp/WEB-INF/conf/dms-extension/dms/dms-templates-configuration.xml`.

Sample configuration:

This below example is configuration for the `nt:file` template, any other template will be put in the same level with this template starting from the line `<object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$NodeType">` as the another node type.

```
<external-component-plugins>
  <target-component>org.exoplatform.services.cms.templates.TemplateService</target-component>
  <component-plugin>
    <name>addTemplates</name>
    <set-method>addTemplates</set-method>
    <type>org.exoplatform.services.cms.templates.impl.TemplatePlugin</type>
    <init-params>
      <value-param>
        <name>autoCreateInNewRepository</name>
        <value>true</value>
      </value-param>
      <value-param>
        <name>storedLocation</name>
        <value>war:/conf/dms-extension/dms/artifacts/templates</value>
      </value-param>
      <value-param>
        <name>repository</name>
        <value>repository</value>
      </value-param>
      <object-param>
        <name>template.configuration</name>
        <description>configuration for the localtion of templates to inject in jcr</description>
        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig">
          <field name="nodeTypes">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$NodeType">
                  <field name="nodetypeName">
                    <string>nt:file</string>
                  </field>
                  <field name="documentTemplate">
                    <boolean>true</boolean>
                  </field>
                  <field name="label">
                    <string>File</string>
                  </field>
                  <field name="referencedView">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$ReferencedView">
                          <field name="templateFile">
                            <string>/file/views/view1.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*</string>
                          </field>
                        </object>
                      </value>
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$ReferencedView">
                          <field name="templateFile">
                            <string>/file/views/admin_view.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*/platform/administrators</string>
                          </field>
                        </object>
                      </value>
                    </collection>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        </collection>
      </field>
      <field name="referencedDialog">
        <collection type="java.util.ArrayList">
          <value>
            <object type="org.exoplatform.services.cms.templates.impl.TemplatePlugin">
              <field name="templateFile">
                <string>/file/dialogs/dialog1.gtmpl</string>
              </field>
              <field name="roles">
                <string>*</string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.templates.impl.TemplatePlugin">
              <field name="templateFile">
                <string>/file/dialogs/admin_dialog.gtmpl</string>
              </field>
              <field name="roles">
                <string>*/platform/administrators</string>
              </field>
            </object>
          </value>
        </collection>
      </field>
      <field name="referencedSkin">
        <collection type="java.util.ArrayList">
          <value>
            <object type="org.exoplatform.services.cms.templates.impl.TemplatePlugin">
              <field name="templateFile">
                <string>/file/skins/Stylesheet-lt.css</string>
              </field>
              <field name="roles">
                <string>*</string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.templates.impl.TemplatePlugin">
              <field name="templateFile">
                <string>/file/skins/Stylesheet-rt.css</string>
              </field>
              <field name="roles">
                <string>*</string>
              </field>
            </object>
          </value>
        </collection>
      </field>
    </object>
  </value>
</collection>
</field>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **name:** *addTemplates*
- **set-method:** *addTemplates*
- **type:** *org.exoplatform.services.cms.templates.impl.TemplatePlugin*
- **Init-params:**

Value-param	Type	Value	Description
autoCreateInNewRepository	boolean	true	Enable the application to import predefined templates at the start-up of template service automatically.
storedLocation	string	war:/conf/dms-extension/dms-locatantsoft-templates	The location of stored templates.
repository	string	repository	Location of stored templates.

- **Object-type:** *org.exoplatform.services.cms.templates.impl.TemplateConfig* that defines all available template files, using the "collection type" configuration.
- **type:** It is the name of each object type. It means the type of template, the further configurations for this type are defined by some specified fields.

Field	Type	Value	Description
nodeTypes	ArrayList	{java.util.ArrayList}	The node type of the template.

- **Object-type:** *org.exoplatform.services.cms.templates.impl.TemplateConfig\$NodeType*

Field	Type	Value	Description
nodetypeName	string	nt:file	The name of template that is saved as a node in system.
documentTemplate	boolean	true	Determine if the node type is a document type.
label	string	file	Visual display of the title for this node.
referencedView	ArrayList	{java.util.ArrayList}	Determine how to display a view.
referencedDialog	ArrayList	{java.util.ArrayList}	Determine how to display a dialog to input information.
referencedSkin	ArrayList	{java.util.ArrayList}	Determine the stylesheet for display.

- **Object type:** *org.exoplatform.services.cms.templates.impl.TemplateConfig\$Template*

Field	Type	Description
templateFile	string	The location of the file store for the template's presentation.
roles	string	Determine who can access this object (View/Dialog/CSS).

3.2.30. XMLdeploymentPlugin

When a site is created, most of end-users want to see something in the page instead of a blank page, so you need this plugin to deploy some "default" contents, such as Banner, Footer, Navigation, Breadcrumb.

There are two main cases to use:

- The site is created only one time when the database is cleaned.
- The site is created at runtime, when a user uses the core features of the GateIn portal.

To use the plugin in the component configuration, you must use the following target-component:

```
<target-component>org.exoplatform.services.deployment.WCMContentInitializerService</target-component>
```

The configuration is applied mainly in *packaging/ecmdemo/webapp/src/main/webapp/WEB-INF/conf/sample-portal/wcm/deployment/acme-deployment-configuration.xml*

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.deployment.WCMContentInitializerService</target-component>
  <component-plugin>
    <name>Content Initializer Service</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.deployment.plugins.XMLDeploymentPlugin</type>
    <description>XML Deployment Plugin</description>
    <init-params>
      <object-param>
        <name>ACME Logo data</name>
        <description>Deployment Descriptor</description>
        <object type="org.exoplatform.services.deployment.DeploymentDescriptor">
          <field name="target">
            <object type="org.exoplatform.services.deployment.DeploymentDescriptor$Target">
              <field name="repository">
                <string>repository</string>
              </field>
              <field name="workspace">
                <string>collaboration</string>
              </field>
              <field name="nodePath">
                <string>/sites content/live/acme/web contents/site artifacts</string>
              </field>
            </object>
          </field>
          <field name="sourcePath">
            <string>war:/conf/sample-portal/wcm/artifacts/site-resources/acme/Logo.xml</string>
          </field>
          <field name="versionHistoryPath">
            <string>war:/conf/sample-portal/wcm/artifacts/site-resources/acme/Logo_versionHistory.zi
          </string>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        <field name="cleanupPublication">
            <boolean>true</boolean>
        </field>
    </object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In which:

- **name:** *Content Initializer Service*
- **set-method:** *addPlugin*
- **type:** *org.exoplatform.services.deployment.plugins.XMLDeploymentPlugin*
- **Object type:** *org.exoplatform.services.deployment.DeploymentDescriptor*

Name	Type	Value	Description
target	Object	org.exoplatform.services.deployment.plugins.XMLDeploymentPlugin (*)	The target node which will contain the imported node.
sourcePath	string	war:/conf/sample-portal/	The absolute path of the XML file.
cleanupPublication	boolean	false	Decide when the publication lifecycle is cleaned up in the target folder after importing the data. true: allow false: not allow
versionHistoryPath	string	war:/conf/sample-portal/	The absolute path of the version history file.

- **Object type:** *org.exoplatform.services.deployment.DeploymentDescriptor\$Target*

Field	Type	Value	Description
repository	string	repository	The repository name of the target node.
workspace	string	collaboration	The collaboration name of the target node.
nodePath	string	/sites content/live/acme/web contents/site	The path of the target node.

Field	Type	Value	Description
		artifacts	

Chapter 4. Developer references

This chapter supply you with the basic knowledge about templates, UI extension, APIs in eXo Content. Through it, you can build your own content. Also, with the step-by-step guides, you can create UI extension, deploy a workflow in eXo Content and do many different actions.

This chapter consists of the main following contents:

- [WCM Templates](#)
- [WCM Explorer](#)
- [Extensions](#)
- [Public REST APIs](#)
- [FAQ](#)

4.1. WCM Templates

4.1.1. Content types

Overview

The templates are applied to a node type or a metadata mixin type. There are two types of templates:

- **Dialogs:** are in the HTML form that allows creating node instances.
- **Views:** are in the HTML fragments which are used to display nodes.

From the ECM admin portlet, the *Manage Template* lists existing node types associated to Dialog and/or View templates. These templates can be attached to permissions (in the usual *membership:group* form), so that a specific one is displayed according to the rights of the user (very useful in a content validation workflow activity).

Document Type

The checkbox defines if the node type should be considered as the **Document type** or not. Content Explorer considers such nodes as user content and applies the following behavior:

- View template will be used to display the document type nodes.
- Document types nodes can be created by the 'Add Document' action.
- Non-document types are hidden (unless the 'Show non document types' option is checked).

Templates are written by using [Groovy Templates](#) that requires some experiences with JCR API and HTML notions.

4.1.1.1. Dialog

Dialogs are Groovy templates that generate forms by mixing static HTML fragments and Groovy calls to the components responsible for building the UI at runtime. The result is a simple but powerful syntax.

4.1.1.1.1. Common parameters

These following parameters are common and can be used for all input fields.

Parameter	Type	Required	Example	Description
jcrPath	string		jcrPath=/node/exo:tt	The relative path inside the current node.
mixintype	string with the commas (,) character.		mixintype= mix:i18n mixintype= mix:votable , mix:commentable , mix:i18n	The list of mixin types you want to initialize when creating the content.
validate	string with the comma (,) character		validate=empty validate=empty, name validate=org.exoplatform.webui.form.validator.	The list of validators you want to apply to the input. Possible values are: <i>name</i> , <i>email</i> , <i>number</i> , <i>empty</i> , <i>null</i> , <i>datetime</i> , <i>length</i> OR validator classes.
editable	string		editable=if-null	The input will be editable only if the value of this parameter is if-null and the value of this input is null or blank.
multiValues	boolean		multiValues=true	Show a multi-valued component if true and must be used only with corresponding multi-valued properties. The default value of this parameter is false.
visible	boolean		visible=true	The input is visible if this value is true.

See also:

- [Text Field](#)
- [Hidden Field](#)
- [Text Area Field](#)
- [Rich Text Field](#)
- [Calendar Field](#)
- [Upload Field](#)
- [Radio Field](#)
- [Select Box Field](#)
- [Checkbox Field](#)
- [Mixin Field](#)
- [Action Field](#)

Note

The *mixintype* can be used only in the root node field (commonly known as the name field).

4.1.1.1.2. Text Field

- **Additional parameters**

See also: [Common parameters](#)

- **Example**

```
<%  
String[] fieldTitle = ["jcrPath=/node/exo:title", "validate=empty"] ;  
uicomponent.addTextField("title", fieldTitle) ;  
%>
```

4.1.1.1.3. Hidden Field

- **Additional parameters**

See also: [Common parameters](#)

- **Example**

```
String[] hiddenField5 = ["jcrPath=/node/jcr:content/dc:date", "visible=false"];  
uicomponent.addCalendarField("hiddenInput5", hiddenField5);
```

4.1.1.1.4. Text Area Field

- Additional parameters

Parameter	Type	Required	Description	Example
rows	Number		The initial text area's number of rows. The value is 10 by default.	rows=20
cols	Number		The initial text area's number of cols. The value is 30 by default .	cols=50

See also: [Common parameters](#)

- Example

```
<%
String[] fieldDescription = ["jcrPath=/node/exo:description", "validate=empty"] ;
uicomponent.addTextAreaField("description", fieldDescription) ;
%>
```

4.1.1.1.5. Rich Text Field

- Additional parameters

Parameter	Type	Required	Description	Example
options	string with the semicolon (;) character		Some options for CKEditor field: toolbar, width and height.	options=CompleteWCM;width:'100%'
toolbar	string		The predefined toolbar for CKEditor. The value can be: Default, Basic, CompleteWCM, BasicWCM, SuperBasicWCM.	options=CompleteWCM
width	string		The width of CKEditor. Its value can be the percent of pixel.	options=width:'100%'
height	string		The height of CKEditor. Its value	options=height:'200px'

Parameter	Type	Required	Description	Example
			can be the percent of pixel.	

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldSummary = ["jcrPath=/node/exo:summary", "options=toolbar:CompleteWCM,width:'100%',height:'200px'"];
uicomponent.addRichTextField("summary", fieldSummary) ;
%>
```

4.1.1.1.6. Calendar Field

- **Additional parameters**

Parameter	Type	Required	Description	Example
options	string		An option for the calendar field: Display time.	options=displaytime

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldPublishedDate = ["jcrPath=/node/exo:publishedDate", "options=displaytime", "validate=datetime"];
uicomponent.addCalendarField("publishedDate", fieldPublishedDate) ;
%>
```

4.1.1.1.7. Upload Field

- **Additional parameters**

See also: [Common parameters](#)

- **Example**

When you create an upload form, you can store an image by two main ways:

- If you want to store the image as a property, use the following code:

```
<%
String[] fieldMedia = ["jcrPath=/node/exo:image"] ;
uicomponent.addUploadField("media", fieldMedia) ;
%>
```

- If you want to store the image as a node, use the following code:

```
<%
String[] hiddenField1 = ["jcrPath=/node/exo:image", "nodetype=nt:resource", "visible=false"] ;
String[] hiddenField2 = ["jcrPath=/node/exo:image/jcr:encoding", "visible=false", "UTF-8"] ;
String[] hiddenField3 = ["jcrPath=/node/exo:image/jcr:lastModified", "visible=false"] ;
uicomponent.addHiddenField("hiddenInput1", hiddenField1) ;
uicomponent.addHiddenField("hiddenInput2", hiddenField2) ;
uicomponent.addHiddenField("hiddenInput3", hiddenField3) ;

String[] fieldMedia = ["jcrPath=/node/exo:image"] ;
uicomponent.addUploadField("media", fieldMedia) ;
%>
```

- But, this code is not complete. If you want to display the **upload** field, the image must be blank, otherwise you can display the image and an action enables you to remove it. You can do as follows:

```
<%
def image = "image";
// If you're trying to edit the document
if(uicomponent.isEditing()) {
    def curNode = uicomponent.getNode();
    // If the image existed
    if (curNode.hasNode("exo:image")) {
        def imageNode = curNode.getNode("exo:image") ;
        // If the image existed and available
        if (imageNode.getProperty("jcr:data").getStream().available() > 0 && (uicomponent.findCo
            def imgSrc = uicomponent.getImage(curNode, "exo:image");
            def actionLink = uicomponent.event("RemoveData", "/exo:image");
            %>
                <div>
                    
                    <a href="$actionLink">
                        
```

4.1.1.1.8. Radio Field

- Additional parameters

Parameter	Type	Required	Description	Example
options	string with the comma characters (.)		Some radio values.	options=radio1,radio2,radio3

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=radio1,radio2,radio3"];
uicomponent.addRadioBoxField("isDeep", fieldDeep);
%>
```

4.1.1.1.9. Select box Field

Parameter	Type	Required	Description	Example
options	string with the comma (,) characters		Some option values.	options=option1,option2,opti

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=checkbox1,checkbox2,checkbox3"];
uicomponent.addCheckBoxField("isDeep", fieldDeep);
%>
```

4.1.1.1.10. Checkbox Field

- **Additional parameters**

Parameter	Type	Required	Description	Example
options	string with the comma (,) characters		Some checkbox values.	options=checkbox1,checkbox2,

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldDeep = ["jcrPath=/node/exo:isDeep", "defaultValues=true", "options=checkbox1,checkbox2,checkbox3"];
uicomponent.addCheckBoxField("isDeep", fieldDeep);
%>
```

4.1.1.1.11. Mixin Field

- **Additional parameters**

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldId = ["jcrPath=/node", "editable=false", "visible=if-not-null"] ;
uicomponent.addMixinField("id", fieldId) ;
%>
```

4.1.1.1.12. Action Field

- **Additional parameters**

Parameter	Type	Required	Description	Example
selectorClass	string		The component to display.	selectorClass=org.exoplatform
selectorIcon	string		The action icon.	selectorIcon>SelectPath24x24

Depending on the selectorClass, some other parameters can be added.

For example, the component `org.exoplatform.ecm.webui.tree.selectone.UIOneNodePathSelector` needs the following parameter:

Parameter	Type	Required	Description	Example
workspaceField	string		The field which enables you to select a workspace.	workspaceField=targetWorkspa

The component `org.exoplatform.ecm.webui.selector.UIPermissionSelector` does not need any special parameters.

See also: [Common parameters](#)

- **Example**

```
<%
String[] fieldPath = ["jcrPath=/node/exo:targetPath", "selectorClass=org.exoplatform.ecm.webui.tree.selectone.
uicomponent.addActionField("targetPath", fieldPath) ;
%>
```

4.1.1.1.13. Interceptors

To add an interceptor to a dialog, you can use this method `uicomponent.addInterceptor(String scriptPath, String type)`

Parameters	Type	Description
scriptPath	string	The relative path to the script file.

Parameters	Type	Description
type	string	The type of interceptor: prev or post.

- **Example**

```
<%
    uicomponent.addInterceptor("ecm-explorer/interceptor/PreNodeSaveInterceptor.groovy", "prev");
%>
```

4.1.1.1.14. How to add a new ECM template with tabs

To avoid refreshing the first tab for every action execution, add a new private function to the template with tabs. In the template, you must insert a new piece of code like the following:

```
private String getDisplayTab(String selectedTab) {
    if ((uicomponent.getSelectedTab() == null && selectedTab.equals("mainWebcontent"))
        || selectedTab.equals(uicomponent.getSelectedTab())) {
        return "display:block";
    }
    return "display:none";
}

private String getSelectedTab(String selectedTab) {
    if (getDisplayTab(selectedTab).equals("display:block")) {
        return "SelectedTab";
    }
    return "NormalTab";
}
```

Changing in every event of **onClick** must be done like the following:

```
<div class="UITab NormalTabStyle">
    <div class="<%=getSelectedTab("mainWebcontent")%>">
        ">
        <div class="LeftTab">
            <div class="RightTab">
                <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "mainWebcontent")%>"><%=_ctx.appRes("w
            </div>
        </div>
    </div>
</div>

<div class="UITab NormalTabStyle">
    <div class="<%=getSelectedTab("illustrationWebcontent")%>">
        ">
        <div class="LeftTab">
            <div class="RightTab">
                <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "illustrationWebcontent")%>"><%=_ctx.a
            </div>
        </div>
    </div>
</div>

<div class="UITab NormalTabStyle">
    <div class="<%= getSelectedTab("contentCSSWebcontent")%>">
        ">
        <div class="LeftTab">
            <div class="RightTab">
                <div class="MiddleTab" onClick="<%=uicomponent.event("ChangeTab", "contentCSSWebcontent")%>"><%=_ctx.app
            </div>
        </div>
    </div>
</div>
```

```
</div>
```

Finally, to display the selected tab, simply add it to the style of `UITabContent` class.

```
<div class="UITabContent" style="<%=getDisplayTab("mainWebcontent")%>">
```

4.1.1.2. View

The following is an sample code of the **View** template of a content node:

- Get a content node to display:

```
<%  
    def node = uicomponent.getNode() ;  
    def originalNode = uicomponent.getOriginalNode()  
%>
```

- Display the name of the content node:

```
<%=node.getName() %>
```

- Display the *exo:title* property of the content node:

```
<%  
    if(node.hasProperty("exo:title")) {  
%>  
    <%=node.getProperty("exo:title").getString() %>  
%>  
    }  
%>
```

- Display the *exo:date* property of the content node in a desired format. For example: "MM DD YYYY" or "YYYY MM DD".

```
<%  
    import java.text.SimpleDateFormat ;  
    SimpleDateFormat dateFormat = new SimpleDateFormat() ;  
%>  
...  
%>  
    if(node.hasProperty("exo:date")) {  
        dateFormat.applyPattern("MMMM dd yyyy") ;  
%>  
        <%=dateFormat.format(node.getProperty("exo:date").getDate().getTime()) %>  
%>  
    }  
%>
```

- Display the translation of the *Sample.view.label.node-name* message in different lanaguages.

```
<%= _ctx.appRes("Sample.view.label.node-name") %>
```

4.1.2. List of Contents

4.1.2.1. Content List Template

The **Content List Template** allows you to view the content list with various templates. eXo Platform supports the following content list templates:

Template	Description
BigHotNewsTemplateCLV.gtmpl	Display contents under one column with a content list. The illustration of each content is displayed above the content.
ContentListViewerDefault.gtmpl	Its function is similar to <i>BigHotNewsTemplateCLV.gtmpl</i> . The illustration of each content is bigger.
DocumentsTemplate.gtmpl	Display contents under a content list with a <i>NodeType</i> icon or the illustration on the left of the corresponding content.
EventsTemplateCLV.gtmpl	Its function is similar to <i>BigHotNewsTemplateCLV.gtmpl</i> , but the illustration of each content is smaller.
OneColumnCLVTemplate.gtmpl	Display contents under one column. The illustration of each content is displayed on its left.
TwoColumnsCLVTemplate.gtmpl	Display contents under two columns. The illustration of each content is displayed on its left.
UIContentListPresentationBigImage.gtmpl	Its function is similar to <i>BigHotNewsTemplateCLV.gtmpl</i> , but the illustration of each content is bigger than the image displayed with <i>ContentListViewerDefault.gtmpl</i> and the text font is different.
UIContentListPresentationDefault.gtmpl	Its function is similar to <i>BigHotNewsTemplateCLV.gtmpl</i> , but the illustration of each content is smaller and the text font is different.
UIContentListPresentationSmall.gtmpl	Display contents under one column with a content list. The images are displayed on the left of the corresponding content and smaller than the images of the other templates.

4.1.2.2. Category Navigation Template

The **Category Navigation Template** displays all contents under the categories.

Template	Description
CategoryList.gtmpl	Display categories as a navigation bar.
CategoryTree.gtmpl	Display categories as a tree.
TagsCloud.gtmpl	Display all tags of the contents.

4.2. WCM Explorer

4.2.1. CSS

- By using WCM, all the stylesheets of each site can be managed online easily. You do not need to access the file system to modify and wait until the server has been restarted. For the structure, each site has its own CSS folder which can contain one or more CSS files. These CSS files have the data, and the priority. If they have the same CSS definition, the higher priority will be applied. You can also disable some of them to make sure the disabled style will no longer be applied into the site.
- For example, a WCM demo package has two sites by default: ACME and Classic. The Classic site has a CSS folder which contains a CSS file called **DefaultStylesheet**. Most of the stylesheets of this site are defined within this stylesheet. Moreover, the ACME site has two CSS files called **BlueStylesheet** and **GreenStylesheet**. The blue one is enabled and the green one is disabled by default. All you need to test is to disable the blue one (by editing it and setting Available to 'false') and enable the green one. Now, back to the homepage and see the magic.

Note

Remember the cache and refresh the browser first if you do not see any changes. Normally, this is the main reason why the new style is not applied.

4.2.2. CKEditor

Basically, if you want to add a rich text area to your dialogs, you can use the [addRichtextField](#) method. However, in case you want to add the rich text editor manually, you first need to use the [addTextAreaField](#) method and some additional Javascripts as shown below:

```
<%
    String[] fieldDescription = ["jcrPath=/node/exo:description"] ;
    uicomponent.addTextAreaField("description", fieldDescription)
%>
<script>
    var instances = CKEDITOR.instances['description'];
    if (instances) instances.destroy(true);
    CKEDITOR.replace('description', {
        toolbar : 'CompleteWCM',
        uiColor : '#9AB8F3'
    });
</script>
```

4.3. Extensions

4.3.1. REST Services

4.3.1.1. Overview

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of "representations" of "resources". A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

At any particular time, a client can either be in transition between application states or "at rest". A client in a REST state is able to interact with its users, but creates no load and consumes no per-client storage on the set of servers or on the network.

The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that may be used the next time, the client chooses to initiate a new state transition.

REST is initially described in the context of HTTP, but is not limited to that protocol. RESTful architectures can be based on other Application Layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximize the use of the pre-existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on its top.

4.3.1.2. Restful Web Service

4.3.1.2.1. HTTP Methods

Here is the convention you should follow:

Method	Definition
GET	Get a Resource. Its state should not be modified.
POST	Create a Resource (or anything that does not fit elsewhere).
PUT	Update a Resource.
DELETE	Delete a Resource.

4.3.1.2.2. Formats

The followings are formats which need to be supported for all your APIs:

- [JSON](#): This format makes developers easy to parse in a lot of languages, such as JavaScript, Python or Ruby.

- [XML](#): Most of Java developers like using this format.
- [ATOM](#): This is a standard format which can be used by many applications.

4.3.1.2.3. Data Format

The default format is JSON.

The response format can be specified by a parameter in the request: *"format"*. You need to specify the format requested.

4.3.1.2.4. REST configuration

First, you need to register the REST service class to the configuration file in the package named *conf.portal*.

```
<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd http://www.exoplaform.org/xml/ns/kernel_1_1.xsd">
  <component>
    <type>org.exoplaform.services.ecm.publication.REST.presentation.document.publication.PublicationGetDocument
  </component>
</configuration>
```

4.3.1.2.5. Create a REST service

You can start creating *GetEditedDocumentRESTService* that implements from the *ResourceContainer* interface as follows:

```
@Path("/presentation/document/edit/")
public class GetEditedDocumentRESTService implements ResourceContainer {
    @Path("/{repository}")
    @GET
    public Response getLastEditedDoc(@PathParam("repository") String repository,
        @QueryParam("showItems") String showItems) throws Exception {
        .....
    }
}
```

Parameters	Definition
@Path("/presentation/document/edit/")	Specify the URI path which a resource or class method will serve requests for.
@PathParam("repository")	Bind the value repository of a URI parameter or a path segment containing the template parameter to a resource method parameter, resource class field, or resource class bean property.
@QueryParam("showItems")	Bind the value showItems of a HTTP query parameter to a resource method parameter, resource class field, or resource class bean property.

4.3.2. UI Extensions

This part describes how to create a sample UI extension.

4.3.2.1. Overview

In eXo Content, it is possible to extend the **Content Explorer** and the **ECM Administration** with the **UI Extension Framework**. Indeed, you can add your own action buttons to the **Content Explorer** and/or add your own managers to the **ECM Administration**.

4.3.2.2. How to add your own tab in ECM Administration

4.3.2.2.1. Add your own UIAction

To add your own UIAction, do as follows:

1. Create a *pom.xml* file with the following content:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:sch
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.exoplatform.ecms</groupId>
    <artifactId>exo-ecms-examples-uiextension-framework</artifactId>
    <version>2.2.0-SNAPSHOT</version>
  </parent>
  <artifactId>exo-ecms-examples-uiextension-framework-manage-wcm-cache</artifactId>
  <name>eXo WCM Cache Examples</name>
  <description>eXo WCM Cache Examples</description>
  <dependencies>
    <dependency>
      <groupId>org.exoplatform.kernel</groupId>
      <artifactId>exo.kernel.container</artifactId>
      <version>2.3.3-GA</version>
      <scope>compile</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.commons</groupId>
      <artifactId>exo.platform.commons.webui.ext</artifactId>
      <version>1.1.4</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.ecms</groupId>
      <artifactId>exo-ecms-core-webui</artifactId>
      <version>2.3.3</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.exoplatform.ecms</groupId>
      <artifactId>exo-ecms-core-webui-administration</artifactId>
      <version>2.3.3</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <build>
    <resources>
      <resource>
        <directory>src/main/java</directory>
        <includes>
          <include>/**/*.xml</include>
        </includes>
      </resource>
      <resource>
        <directory>src/main/resources</directory>
        <includes>
          <include>/**/*.properties</include>
          <include>/**/*.xml</include>
          <include>/**/*.jar</include>
          <include>/**/*.pom</include>
          <include>/**/*.conf</include>
          <include>/**/*.gtmpl</include>
          <include>/**/*.gif</include>
          <include>/**/*.jpg</include>
          <include>/**/*.png</include>
        </includes>
      </resource>
    </resources>
  </build>
</project>
```

```

        </includes>
      </resource>
    </resources>
  </build>
</project>

```

2. Create the `src/main/java` directory and start launching `mvn eclipse:eclipse`. Then, you can launch your eclipse and import this new project.

3. Create a new class called `org.exoplatform.wcm.component.cache.UIWCMCacheComponent` that extends `org.exoplatform.ecm.webui.component.admin.manager.UIAbstractManagerComponent`.

4.3.2.2.2. Add your own ActionListener

With the Webui framework, you will be notified if any given action is triggered. You just need to call your own action listener (`$ACTIONNAME`) **ActionListener**. For example, to create your own action listener for **CacheView**, do as follows:

1. Call **CacheViewActionListener**.

2. Add a static inner class called `CacheViewActionListener` that extends `org.exoplatform.ecm.webui.component.admin.listener.UIECMAdminControlPanelActionListener`.

You will see the expected code below:

```

public class CacheViewComponent extends UIAbstractManagerComponent {
    public static class CacheViewActionListener extends UIECMAdminControlPanelActionListener
    <UIWCMCacheComponent>{
        public void processEvent(Event
        <UIWCMCacheComponent>event) throws Exception {
            UIECMAdminPortlet portlet = event.getSource().getAncestorOfType(UIECMAdminPortlet.class);
            UIECMAdminWorkingArea uiWorkingArea = portlet.getChild(UIECMAdminWorkingArea.class);
            uiWorkingArea.setChild(UIWCMCachePanel.class);
            event.getRequestContext().addUIComponentToUpdateByAjax(uiWorkingArea);
        }
    }
}

```

3. Create the `src/main/java` directory and launch `mvn eclipse:eclipse`. You can then launch your eclipse and import this new project.

4. Create a new configuration file called `conf/portal/configuration.xml` to register your action (see [Register your UI Action](#)) with the `org.exoplatform.webui.ext.UIExtensionManager` service.

4.3.2.2.3. Register your UI Action

To register your UI Action, do as follows:

1. Create the code:

```

<configuration xmlns="http://www.exoplaform.org/xml/ns/kernel_1_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  <external-component-plugins>
    <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
    <component-plugin>
      <name>Add.Actions</name>
      <set-method>registerUIExtensionPlugin</set-method>
      <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
      <init-params>
        <object-param>
          <name>CacheView</name>

```

```

        <object type="org.exoplatform.webui.ext.UIExtension">
          <field name="type">
            <string>org.exoplatform.ecm.dms.UIECMAdminControlPanel</string>
          </field>
          <field name="name">
            <string>CacheView</string>
          </field>
          <field name="category">
            <string>GlobalAdministration</string>
          </field>
          <field name="component">
            <string>org.exoplatform.wcm.component.cache.UIWCMCacheComponent</string>
          </field>
        </object>
      </object-param>
    <object-param>
      <name>UIWCMCacheManager</name>
      <object type="org.exoplatform.webui.ext.UIExtension">
        <field name="type">
          <string>org.exoplatform.ecm.dms.UIECMAdminControlPanel</string>
        </field>
        <field name="name">
          <string>UIWCMCacheManager</string>
        </field>
        <field name="category">
          <string>GlobalAdministration</string>
        </field>
        <field name="component">
          <string>org.exoplatform.wcm.manager.cache.UIWCMCacheManagerComponent</string>
        </field>
        <field name="extendedFilters">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.webui.ext.filter.impl.UserACLFilter">
                <field name="permissions">
                  <collection item-type="java.lang.String" type="java.util.ArrayList">
                    <value>
                      <string>*:/platform/administrators</string>
                    </value>
                  </collection>
                </field>
              </object>
            </value>
          </collection>
        </field>
      </object>
    </object-param>
  </init-params>
</component-plugin>
</external-component-plugins>
</configuration>

```

Note

With this configuration, only the users with the `*:/platform/administrators` membership have the right to access the CacheView item.

2. Launch **mvn clean install**.

3. Copy the resource bundle.

Note

All resources can be located in the `src/main/resource` package because the resources (*.xml, images, conf file) and the code are separated. This is very useful in a hierarchical structure.

Create `ExamplePortlet_en.xml` with the following content and add it to the `src/main/resource` package:

```

<bundle>
  <!-- ##### # org.exoplatform.wcm.
  # ##### -->

  <UIWCMCacheForm>
    <action>
      <Cancel>Cancel</Cancel>
      <Save>Save</Save>
      <Clear>Clear the cache</Clear>
    </action>
    <label>
      <maxsize>Max size :</maxsize>
      <livetime>Live time in sec :</livetime>
      <isCacheEnable>Cache enabled(should always be on production enviroment)</isCacheEnable>
      <hit>Hit count :</hit>
      <currentSize>Current size</currentSize>
      <miss>Miss count :</miss>
    </label>
  </UIWCMCacheForm>

  <!-- ##### # org.exoplatform.wcm.
  # ##### -->

  <UIWCMCacheManagerForm>
    <action>
      <Cancel>Cancel</Cancel>
      <Save>Save</Save>
      <Clear>Clear the cache</Clear>
    </action>
    <label>
      <cacheModify>Cache to modify :</cacheModify>
      <maxsize>Max size :</maxsize>
      <livetime>Live time in sec :</livetime>
      <isCacheEnable>Cache enabled(should always be on production enviroment)</isCacheEnable>
      <hit>Hit count :</hit>
      <currentSize>Current size</currentSize>
      <miss>Miss count :</miss>
    </label>
  </UIWCMCacheManagerForm>

  <UIECMAdminControlPanel>
    <tab>
      <label>
        <GlobalAdministration>Global Administration</GlobalAdministration>
      </label>
    </tab>
    <label>
      <UIWCMCache>WCM Cache</UIWCMCache>
      <UIWCMCachePanel>WCM Cache Administration</UIWCMCachePanel>
      <UIWCMCacheManager>Managing Caches</UIWCMCacheManager>
      <UIWCMCacheManagerPanel>WCM Cache Management</UIWCMCacheManagerPanel>
    </label>
  </UIECMAdminControlPanel>
</bundle>

```

You must add the following content to *configuration.xml* to register the resource bundle.

Note

By being added this configuration, the resource bundle has been completely separated from the original system. This is clearly useful for you to get an independent plugin.

```

<external-component-plugins>
  <!-- The full qualified name of the ResourceBundleService -->
  <target-component>org.exoplatform.services.resources.ResourceBundleService</target-component>
  <component-plugin>
    <!-- The name of the plugin -->
    <name>ResourceBundle Plugin</name>
    <!-- The name of the method to call on the ResourceBundleService in order to register the ResourceBundleService -->
    <set-method>addResourceBundle</set-method>
    <!-- The full qualified name of the BaseResourceBundlePlugin -->

```

```
<type>org.exoplatform.services.resources.impl.BaseResourceBundlePlugin</type>
<init-params>
  <values-param>
    <name>init.resources</name>
    <description>Store the following resources into the db for the first launch</description>
    <value>locale.portlet.cache.ExamplePortlet</value>
  </values-param>
  <values-param>
    <name>portal.resource.names</name>
    <description>The properties files of the portal , those file will be merged
      into one ResoruceBundle properties
    </description>
    <value>locale.portlet.cache.ExamplePortlet</value>
  </values-param>
</init-params>
</component-plugin>
</external-component-plugins>
```

4.3.2.2.4. Run your own UI extension sample

Do as follows:

1. Run Tomcat.
2. Sign in as "root".
3. Go to the ECM Administration. You will see that a new extension has been already added to ECM Administrator.

4.3.3. Authoring Extension

Publication add-ons for eXo Content.

4.3.3.1. Extended Publication Plugin

4.3.3.1.1. States

This extended publication has new states and new profiles that are enabled in eXo Content.

- Profiles
 - Author: This profile can edit a content and mark this content as redacted.
 - Approver: This profile approves a pending content (marked by the Author).
 - Publisher: This profile publishes contents or marks them as "Ready for publication" in multi-server mode.
 - Archiver: An administrative profile which moves contents to an archive storage.
- States
 - enrolled: It is a pure technical state, generally used for content creation.
 - draft (Author): Content is in editing phase.
 - pending (Author): The author validates the content.
 - approved (Approver): A content is approved by the manager.

- **inreview (Manager):** This state can be used when a second approval state is needed (for i18n translation for example).
- **staged (Publisher):** A content is ready for publication (multi-server mode).
- **published (Publisher or Automatic):** A content is published and visible in the **Live** mode.
- **unpublished (Publisher or Automatic):** A content is not visible in the **Live** mode.
- **obsolete:** A content can still be published but it is not in an editing lifecycle anymore.
- **archived (Automatic):** A content is archived and ready to be moved in the archive workspace if enabled.

4.3.3.1.2. Start/End publication dates

In most cases, you do not want to publish a content directly, but at a defined date and you can also want the content to be unpublished automatically after that. New properties are added to the new publication plugin, that allows you to manage this:

- *publication:startPublishedDate_*
- *publication:endPublishedDate_*

The eXo Content rendering engine does not know anything about publication dates, so another service needs to manage that. When the publisher sets start/end publication dates, he can "stage" the content. The content will go automatically to the "published" state when the start date arrives and to the "unpublished" state after end date. A cron job checks every hour (or less) all contents which need to be published (the start date in the past and the "staged" state) or unpublished (the end date in the past and the "published" state).

Thus, the publication dates are not mandatory and a content can go to:

- **Staged:** in multi-server mode, the publisher can only put the content to the "staged" state and wait for auto-publication.
- **Published:** in single-server mode, the publisher can directly publish a content (with or without publication dates).

4.3.3.1.3. New Publication Mixin

```
<nodeType hasOrderableChildNodes="false" isMixin="true" name="publication:authoringPublication" primaryItemName=  
  <supertypes>  
    <supertype>publication:stateAndVersionBasedPublication</supertype>  
  </supertypes>  
  <propertyDefinitions>  
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="publication:startPublishedDate_>  
      <valueConstraints/>  
    </propertyDefinition>  
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="publication:endPublishedDate_>  
      <valueConstraints/>  
    </propertyDefinition>  
  </propertyDefinitions>  
</nodeType>
```

Publication plugin UI:

Note that some labels containing special or non-ASCII characters could not be well displayed in the publication

UI. You can extend the width of the current UI State button by adding:

```
.UIPublicationPanel .StatusTable .ActiveStatus {  
  width: 75px !important;  
}
```

Also, for the publication date inputs, *UIPublicationPanel* should not initialize the dates to any default value. The publishing and unpublish CRON jobs will do this:

- A staged document with null publication start date is published instantly.
- A document with null publication end date is published forever.

See the export section for more information about the CRON jobs.

4.3.3.2. Publication Manager

The Publication Manager manages lifecycles and contexts in the eXo Content platform. It allows to manages different lifecycles based on different publication plugin in the platform.

```
public interface PublicationManager {  
    public List<Lifecycle> getLifecycles();  
    public List<Context> getContexts();  
    public Context getContext(String name);  
    public Lifecycle getLifecycle(String name);  
    public List<Lifecycle> getLifecyclesFromUser(String remoteUser, String state);  
}
```

In which:

- *getLifecycles*: returns a list of lifecycles (see below), with lifecycle name, publication plugin involved and possible states.
- *getContexts*: returns a list of context, with name, related Lifecycle and other properties (see below).
- *getContext*: returns a context by its name.
- *getLifecycle*: returns a lifecycle by its name.
- *getLifecycleFromUser*: returns a list of lifecycles in which the user has rights (based on membership property).

4.3.3.2.1. Lifecycle

A lifecycle is defined by a simple vertical workflow with steps (states) and profiles (membership). Each lifecycle is related to a **Publication** plugin (compliant with the JBPM or Bonita business processes).

For example: *Two lifecycles with/without states*

```
<external-component-plugins>  
  <target-component>org.exoplatform.services.wcm.publication.PublicationManager</target-component>
```

```

<component-plugin>
  <name>AddLifecycle</name>
  <set-method>addLifecycle</set-method>
  <type>org.exoplatform.services.wcm.publication.lifecycles.StatesLifecyclePlugin</type>
  <init-params>
    <object-param>
      <name>lifecycles</name>
      <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.LifecyclesConfig">
        <field name="lifecycles">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
                <field name="name">
                  <string>lifecycle1</string>
                </field>
                <field name="publicationPlugin">
                  <string>States and versions based publication</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
                <field name="name">
                  <string>lifecycle2</string>
                </field>
                <field name="publicationPlugin">
                  <string>Authoring publication</string>
                </field>
                <field name="states">
                  <collection type="java.util.ArrayList">
                    <value>
                      <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.State">
                        <field name="state">
                          <string>draft</string>
                        </field>
                        <field name="memberships">
                          <collection type="java.util.ArrayList">
                            <value>
                              <string>author:/CA/communicationDG</string>
                            </value>
                            <value>
                              <string>author:/CA/alerteSanitaire</string>
                            </value>
                            <value>
                              <string>author:/CA/alerteInformatique</string>
                            </value>
                            <value>
                              <string>author:/CA/informations</string>
                            </value>
                          </collection>
                        </field>
                      </object>
                    </value>
                  </collection>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
                <field name="state">
                  <string>pending</string>
                </field>
                <field name="membership">
                  <string>author:/platform/web-contributors</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
                <field name="state">
                  <string>approved</string>
                </field>
                <field name="membership">
                  <string>manager:/platform/web-contributors</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
                <field name="state">
                  <string>staged</string>
                </field>
                <field name="membership">

```

```

        <string>publisher:/platform/web-contributors</string>
      </field>
    </object>
  </value>
</value>
<value>
  <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
    <field name="state">
      <string>published</string>
    </field>
    <field name="membership">
      <string>automatic</string>
    </field>
  </object>
</value>
</collection>
</field>
</object>
</value>
<value>
  <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
    <field name="name">
      <string>lifecycle3</string>
    </field>
    <field name="publicationPlugin">
      <string>Authoring publication</string>
    </field>
    <field name="states">
      <collection type="java.util.ArrayList">
        <value>
          <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
            <field name="state">
              <string>draft</string>
            </field>
            <field name="membership">
              <string>author:/platform/web-contributors</string>
            </field>
          </object>
        </value>
        <value>
          <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.Lifecycle">
            <field name="state">
              <string>published</string>
            </field>
            <field name="memberships">
              <collection type="java.util.ArrayList">
                <value>
                  <string>publisher:/CA/communicationDG</string>
                </value>
                <value>
                  <string>publisher:/CA/alerteSanitaire</string>
                </value>
                <value>
                  <string>publisher:/CA/alerteInformatique</string>
                </value>
                <value>
                  <string>publisher:/CA/informations</string>
                </value>
              </collection>
            </field>
          </object>
        </value>
      </collection>
    </field>
  </object>
</value>
</collection>
</field>
</object>
</value>
</collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

In the last example, there are three lifecycles:

- Lifecycle 1: Based on [StatesAndVersionsPublicationPlugin](#) .
 - This allows to be backward compliant with older eXo Content releases. If all your site contents are using an existing plugin, you can create a lifecycle for it and it will work.
 - For new instances, you should use the new plugin with dynamic states capabilities.
- Lifecycle 2: Based on [AuthoringPublicationPlugin](#) .
 - Visibility: Define only the "visible" steps. In this example, there is no step for "enrolled". Even if this step exists, it will not be displayed in the UI.
 - Automatic: Set a step as "automatic". In this mode, the step will be visible in the UI but it will be managed by the system (e.g. a cron job).
- Lifecycle 3: Simulates the *StatesAndVersionsPublicationPlugin* plugin. Note that this simple lifecycle will work in a single server configuration.

4.3.3.2.1.1. Listen to a lifecycle

When a state is changed, you can broadcast an event to add features. The event could look like this:

```
listenerService.broadcast(AuthoringPlugin.POST_UPDATE_STATE_EVENT, null, node);
```

Listener declaration could look like this:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.listener.ListenerService</target-component>
  <component-plugin>
    <name>PublicationService.event.postUpdateState</name>
    <set-method>addListener</set-method>
    <type>org.exoplatform.services.wcm.publication.listener.post.PostUpdateStateEventListener</type>
    <description>this listener will be called every time a content changes its current state</description>
  </component-plugin>
</external-component-plugins>
```

4.3.3.2.2. Context

A context is defined by simple rules. In eXo Content, you can select to enroll the content in a specific lifecycle (for example, publication plugin) based on context parameters. There are three parameters used to define contexts:

- Remote User: The current user who can create/edit the content.
- Current site name: The site from where the content is created (not the storage but the navigation).
- Node: The node which you want to enroll.

From these parameters, you can easily connect and define contexts based on:

- Membership: Does the current user have this membership?
- Site: On this particular site, you want to enroll contents in a specific lifecycle.

- Path: You can enroll contents in the lifecycles based on their path (from the Node).
- Type of content: You can enroll contents in the lifecycles based on their nodetype (from the Node).

Because each site has a content storage (categories + physical storage), you can select the right lifecycle for the right storage/site. To avoid conflicts on contexts, you can set a priority (the less is the best).

For example, **Different Contexts**:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.publication.PublicationManager</target-component>
  <component-plugin>
    <name>AddContext</name>
    <set-method>addContext</set-method>
    <type>org.exoplatform.services.wcm.publication.context.ContextPlugin</type>
    <init-params>
      <object-param>
        <name>contexts</name>
        <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig">
          <field name="contexts">
            <collection type="java.util.ArrayList">
              <value>
                <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig">
                  <field name="name">
                    <string>contextdefault</string>
                  </field>
                  <field name="priority">
                    <string>200</string>
                  </field>
                  <field name="lifecycle">
                    <string>lifecycle1</string>
                  </field>
                </object>
                <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig">
                  <field name="name">
                    <string>context1</string>
                  </field>
                  <field name="priority">
                    <string>100</string>
                  </field>
                  <field name="lifecycle">
                    <string>lifecycle1</string>
                  </field>
                  <field name="membership">
                    <string>*:/platform/web-contributors</string>
                  </field>
                  <field name="site">
                    <string>acme</string>
                  </field>
                  <field name="path">
                    <string>repository:collaboration:/sites content/live/acme/categories</string>
                  </field>
                </object>
                <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig">
                  <field name="name">
                    <string>context2</string>
                  </field>
                  <field name="priority">
                    <string>100</string>
                  </field>
                  <field name="lifecycle">
                    <string>lifecycle1</string>
                  </field>
                  <field name="site">
                    <string>classic</string>
                  </field>
                </object>
                <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig">
                  <field name="name">
                    <string>context3</string>
                  </field>
                  <field name="priority">
                    <string>80</string>
                  </field>
                </object>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        <field name="lifecycle">
          <string>lifecycle3</string>
        </field>
        <field name="membership">
          <string>manager:/company/finances</string>
        </field>
        <field name="path">
          <string>repository:collaboration:/documents/company/finances</string>
        </field>
      </object>
    <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConf
      <field name="name">
        <string>context4</string>
      </field>
      <field name="priority">
        <string>50</string>
      </field>
      <field name="lifecycle">
        <string>lifecycle4</string>
      </field>
      <field name="memberships">
        <collection type="java.util.ArrayList">
          <value>
            <string>manager:/CA/communicationDG</string>
          </value>
          <value>
            <string>manager:/CA/alerteSanitaire</string>
          </value>
          <value>
            <string>manager:/CA/alerteInformatique</string>
          </value>
          <value>
            <string>manager:/CA/informations</string>
          </value>
        </collection>
      </field>
      <field name="path">
        <string>repository:collaboration:/documents/company/finances</string>
      </field>
      <field name="nodetype">
        <string>exo:article</string>
      </field>
    </object>
  </value>
</collection>
</field>
</object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

The logic is very simple. When creating a content, it should be attached a lifecycle with the lifecycle priority:

- *context4* is the most important (priority=50): you will enroll the content in the lifecycle "lifecycle4" if:
 - The content creator has the *manager:/company/finances* membership.
 - The content is stored in *repository:collaboration:/documents/company/finances* or any subfolders.
 - The content is a 'exo:article'.
- If not, you will continue with *context3*.

The logic is very simple. When you create a content, go lifecycle by lifecycle starting with the better priority:

- *context4* is the most important (priority=50): you will enroll the content in the lifecycle "lifecycle4" if:
 - The content creator has the *manager:/company/finances* membership.

- The content is stored in *'repository:collaboration:/documents/company/finances'* or any subfolders.
- The content is a *exo:article_*.
- If not, you will continue with *context3*.

Note

The contexts will be used only when the content is created and when you want to enroll it in a lifecycle for the first time. Once you have the corresponding lifecycle, you will set the lifecycle inside the content (see [New Authoring Mixin](#)) and the context service will not be called again for this content.

4.3.3.2.3. New Authoring Mixin

```
<nodeType hasOrderableChildNodes="false" isMixin="true" name="publication:authoring" primaryItemName="">
  <propertyDefinitions>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="publication:lastUser" onPar
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="publication:lifecycle" onPa
      <valueConstraints/>
    </propertyDefinition>
  </propertyDefinitions>
</nodeType>
```

When adding the content in a lifecycle, set the *publication:lifecycle_* property with the corresponding lifecycle.

Note

A content can be in one lifecycle only.

Each time you change from one state to another, set the user who changed the state in *publication:lastUser*.

Querying based on publication status:

By adding this mixin to contents, you can access contents by simple queries based on the current user profile. For example:

- All your draft contents:
 - query: `select * from nt:base_ where publication:currentState"draft" and publication:lastUser="benjamin".`
- All the contents you have to approve.
 - call: `PublicationManager.getLifecycles('benjamin','approved')` => returns lifecycles where you can go to the 'approved' state.
 - query: `select * from nt:base_ where publication:currentState="pending" and publication:lifecycle="lifecycle1" or publication:lifecycle="lifecycle3".`
- All the content that will be published tomorrow.

- query: select * from nt:base_ where publication:currentState="staged" and publication:startPublishedDate_="xxxx".

4.4. Public REST APIs

4.4.1. ThumbnailRESTService

Service name	Service URL	Location	Description
ThumbnailRESTService	{portalname}/{restcontextname}/thumbnailImage/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-services	Return a responding data as a thumbnail image.

Note

- {portalname}: The name of the portal.
- {restcontextname}: The context name of rest webapplication which is deployed to the "{portalname}" portal.

• APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
getThumbnailImage	{portalname}/{restcontextname}/thumbnailImage/medium/{repoName}/{workspaceName}/{nodePath}		repoName workspaceName nodePath	String string string	Return an image with the medium size (64x64).
getLargeImage	{portalname}/{restcontextname}/thumbnailImage/large/{repoName}/{workspaceName}/{nodePath}		repoName workspaceName nodePath	String string string	Return an image with the large size (300x300).
getSmallImage	{portalname}/{restcontextname}/thumbnailImage/small/{repoName}/{workspaceName}/{nodePath}		repoName workspaceName nodePath	String string string	Return an image with the small size (32x32).
					105

Name	Service endpoint	URL	Parameters	Values	Description
			nodePath	string	
getCustomImage	{portalname}/{restcontextname}/thumbnailImage/custom/{size}	{portalname}/{restcontextname}/thumbnailImage/custom/{size}	repoName workspaceName nodePath	String string string string	Return an image with the custom size.
getOriginImage	{portalname}/{restcontextname}/thumbnailImage/origin/{repoName}/{workspaceName}/{nodePath}	{portalname}/{restcontextname}/thumbnailImage/origin/{repoName}/{workspaceName}/{nodePath}	repoName workspaceName nodePath	String string string	Return an image with the original size.

4.4.2. RssConnector

Service name	Service URL	Location	Description
RssConnector	{portalname}/{restcontextname}/feed/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Generate an RSS feed.

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
generate	{portalname}/{restcontextname}/feed/rss/	{portalname}/{restcontextname}/feed/rss/	repository workspace server siteName	string string string string	Generate an RSS feed.

Name	Service endpoint	URL	Parameters	Values	Description
			title desc folderPath orderBy orderType lang detailPage detailParam recursive	string string string string string string string string	

4.4.3. FCKCoreRESTConnector

Service name	Service URL	Location	Description
FCKCoreRESTConnector	{portalname}/{restcontextname}	{portalname}/{restcontextname}//fckconnector/jcr Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Get a list of files and folders, and create a folder and upload files.

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
getFoldersAndFiles	{portalname}/{restcontextname}	{portalname}/{restcontextname}/fckconnector/jcr/getFoldersAndFiles/{repositoryName}	repositoryName workspaceName currentFolder	string string string	Return the folders and the files in the current folder.

Name	Service endpoint	URL	Parameters	Values	Description
			command	string	
			type	string	
createFolder	{portalname}/{restcontextname}/fckconnector/jcr/createFolder	{portalname}/{restcontextname}/fckconnector/jcr/createFolder	repositoryName	string	Create a folder under the current folder.
			workspaceName	string	
			currentFolder	string	
			newFolderName	string	
			language	string	
uploadFile	{portalname}/{restcontextname}/fckconnector/jcr/uploadFile	{portalname}/{restcontextname}/fckconnector/jcr/uploadFile		HttpServletResponse	Upload a file with the <i>HttpServletRequest</i> .
processUpload	{portalname}/{restcontextname}/fckconnector/jcr/uploadFile	{portalname}/{restcontextname}/fckconnector/jcr/uploadFile	repositoryName	string	Control the process of uploading a file, such as aborting, deleting or progressing the file.
			workspaceName	string	
			currentFolder	string	
			action	string	
			language	string	
			fileName	string	
			uploadId	string	

4.4.4. ResourceBundleConnector

Service name	Service URL	Location	Description
ResourceBundleConnector	{portalname}/{restcontextname}/bundle/	Maven groupId: org.exoplatform.ecms ArtifactId:	Get the bundle basing on the key and the locale.

Service name	Service URL	Location	Description
		exo-ecms-core-connector	

- APIs usage:

Name	Service endpoint URL	Parameters	Values	Description
getBundle	{portalname}/{restcontextname}/bundle/getBundle/{key}/{locale}	key locale	string string	Get the bundle basing on the key and the locale.

4.4.5. VoteConnector

Service name	Service URL	Location	Description
VoteConnector	{portalname}/{restcontextname}/contents/vote/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Return and set a vote value of a given node in the sent parameter.

- APIs usage:

Name	Service endpoint URL	Parameters	Values	Description
postVote	{portalname}/{restcontextname}/contents/vote/postVote/{repositoryName}	workspaceName jcrPath vote lang	string string string string string	Set a vote value for a given content.
getVote	{portalname}/{restcontextname}/contents/vote/getVote/{repositoryName}		string	Return a vote value for a given content.

Name	Service endpoint	URL	Parameters	Values	Description
			workspaceName	string	
			jcrPath	string	

4.4.6. DriverConnector

Service name	Service URL	Location	Description
DriverConnector	{portalname}/{restcontextname}/wcmDriver/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Return a drive list, a folder list and a document list in a specified location for a given user. Also, it processes the file uploading action.

- **APIs usage:**

Name	Service endpoint	URL	Parameters	Values	Description
getDrivers	{portalname}/{restcontextname}/wcmDriver/getDrivers/			string	Return a list of drives for the current user.
getFoldersAndFiles	{portalname}/{restcontextname}/wcmDriver/getFoldersAndFiles/	driverName		string	Return all folders and files in a given location.
		currentFolder		string	
		currentPortal		string	
		repositoryName		string	
		workspaceName		string	
		filterBy		string	
uploadFile	{portalname}/{restcontextname}/wcmDriver/uploadFile/uploadFile/			string	Uploads a file.
processUpload	{portalname}/{restcontextname}/wcmDriver/uploadFile/controlUpload/	repositoryName		string	Control the process of uploading a file, such as aborting, deleting or processing the file.
		workspaceName		string	

Name	Service endpoint	URL	Parameters	Values	Description
			driverName	string	
			currentFolder	string	
			currentPortal	string	
			userId}	string	
			jcrPath	string	
			action	string	
			language	string	
			fileName	string	
			uploadId	string	

4.4.7. GadgetConnector

Service name	Service URL	Location	Description
GadgetConnector	{portalname}/{restcontextname}	{portalname}/wcmGadget/ Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Instantiate a new gadget connector.

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
getFoldersAndFiles	{portalname}/{restcontextname}	{portalname}/wcmGadget/getFoldersAndFiles/{lang}/{currentFolder}	currentFolder	string	Get the folders and files.
			lang	string	
			host	string	

4.4.8. PortalLinkConnector

Service name	Service URL	Location	Description
PortalLinkConnector	{portalname}/{restcontextname}/portalLinks/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-connector	Return a page URI for a given location.

- APIs usage:

Name	Service endpoint	Parameters	Values	Description
getPageURI	{portalname}/{restcontextname}/portalLinks/getFoldersAndFiles	currentFolder	string string string	Get the page URI.

4.4.9. GetEditedDocumentRESTService

Service name	Service URL	Location	Description
GetEditedDocumentRESTService	{portalname}/{restcontextname}/presentation/documents	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-publication	Return the latest edited documents.

- APIs usage:

Name	Service endpoint	Parameters	Values	Description
getLastEditedDoc	{portalname}/{restcontextname}/presentation/document/edit		string	Return the latest edited documents.

4.4.10. PublicationGetDocumentRESTService

Service name	Service URL	Location	Description
PublicationGetDocumentRESTService	{portalname}/{restcontextname}/publication/presentation/{repositoryId}	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-publication	Return a list of published documents.

• APIs usage:

Name	Service URL	Parameters	Values	Description
getPublishDocument	{portalname}/{restcontextname}/publication/presentation/{repositoryId}	workspace state showItems	string string string	Return a list of published document by the default plugin.
getPublishedListDocument	{portalname}/{restcontextname}/publication/presentation/{repositoryId}	workspace publicationPluginName state showItems	string string string string	Return a list of published documents by a specific plugin.

4.4.11. FavoriteRESTService

Service name	Service URL	Location	Description
FavoriteRESTService	{portalname}/{restcontextname}/favorite/{userId}	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-services	Return a list of favourite documents of a given user.

Service name	Service URL	Location	Description

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
getFavoriteByUser	{portalname}/{restcontextname}/favorite/all/{repoName}/{workspaceName}/{userName}		repoName workspaceName userName showItems	string string string string	Return a list of favourite documents of a given user.

4.4.12. RESTImagesRendererService

Service name	Service URL	Location	Description
RESTImagesRendererService	{portalname}/{restcontextname}/images/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-core-services	Get the image binary data of a given image node.

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
serveImage	{portalname}/{restcontextname}/images/{repositoryName}/{workspaceName}/{nodeIdentifier}		repositoryName workspaceName nodeIdentifier param	string string string string	Get the image binary data of a given image node.

4.4.13. LifecycleConnector

Service name	Service URL	Location	Description
LifecycleConnector	{portalname}/{restcontextname}/authoring/	Maven groupId: org.exoplatform.ecms ArtifactId: {exo-ecms-ext-authoring-services}}	Return a list of contents in a given state range of the publication lifecycle.

• **APIs usage:**

Name	Service URL endpoint	Parameters	Values	Description
byState	{portalname}/{restcontextname}/authoring/bystate/	fromstate user lang workspace json	string string string string string	Return a list of contents from the given state to the last state.
toState	{portalname}/{restcontextname}/authoring/toState/	fromstate tostate user lang workspace json	string string string string string	Return a list of contents from the beginning state to the last state.
byDate	{portalname}/{restcontextname}/authoring/byDate/	fromstate date lang workspace	string string string string	Return a list of contents from the given beginning state and published before the given date.
				115

Name	Service endpoint	URL	Parameters	Values	Description
			json	string string	

4.4.14. CopyContentFile

Service name	Service URL	Location	Description
CopyContentFile	{portalname}/{restcontextname}/copyfile/	Maven groupId: org.exoplatform.ecms ArtifactId: exo-ecms-ext-authoring-services	Copy a file.

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
copyFile	{portalname}/{restcontextname}/copyfile/		parameters: the file list, copy information		Copy a file.

4.4.15. PDFViewerRESTService

Service name	Service URL	Location	Description
PDFViewerRESTService	{portalname}/{restcontextname}/pdfviewer//{repoName}/{workspaceName}/pdf/{pageNumber}	Maven groupId: org.exoplatform.ecms ArtifactId: ecms-core-viewer	Return the pdf content to display on the web page

- APIs usage:

Name	Service endpoint	URL	Parameters	Values	Description
getCoverImage	{portalname}/{restcontextname}/pdfviewer//{repoName}/{workspaceName}/pdf/{pageNumber}		repoName workspaceName	string string	Return the thumbnail image for a pdf document.

Name	Service endpoint	URL	Parameters	Values	Description
			uuid	string	
			pageNumber	string	
			rotation	string	
			scale		

4.5. Public Java APIs

4.5.1. TaxonomyService

Taxonomy service is used to work with taxonomies. In this service, there are many functions which enable you to add, find, or delete taxonomies from a node.

Method	Param	Return	Description
getTaxonomyTree (String repository, String taxonomyName, boolean system) throws RepositoryException;	repository: The name of repository. taxonomyName: The name of the taxonomy. system: Indicates whether the nodes must be retrieved using a session system or user session.	node	Return the root node of the given taxonomy tree.
getTaxonomyTree (String repository, String taxonomyName) throws RepositoryException;	repository: The name of repository. taxonomyName: The name of the taxonomy.	node	Return the root node of the given taxonomy tree with the user session.
getAllTaxonomyTrees (String repository, boolean system) throws RepositoryException;	repository: The name of repository. system: Indicates whether the nodes must be retrieved using a	List<Node>	Return the list of all the root nodes of the taxonomy tree available.

Method	Param	Return	Description
	session system or user session.		
getAllTaxonomyTrees (String repository) throws RepositoryException;	repository: The name of repository.	List<Node>	Return the list of all the root nodes of the taxonomy tree available with the user session.
hasTaxonomyTree (String repository, String taxonomyName) throws RepositoryException;	repository: The name of repository. taxonomyName: The name of the taxonomy.	boolean	Check if a taxonomy tree with the given name has already been defined.
addTaxonomyTree (Node taxonomyTree) throws RepositoryException, TaxonomyAlreadyExistsException;	taxonomyTree: The taxonomy tree to define.	void	Define a node as a new taxonomy tree.
updateTaxonomyTree (String taxonomyName, Node taxonomyTree) throws RepositoryException;	taxonomyName: The name of the taxonomy to update. taxonomyTree: The taxonomy tree to define.	void	Re-define a node as a taxonomy tree.
removeTaxonomyTree (String taxonomyName) throws RepositoryException	taxonomyName: The name of the taxonomy to remove.	void	Remove the taxonomy tree definition.
addTaxonomyNode (String repository, String workspace, String parentPath, String taxoNodeName, String creator) throws RepositoryException, TaxonomyNodeAlreadyExistsException;	repository: The name of the repository. workspace: The name of the workspace parentPath: The place where the taxonomy node will be added. taxoNodeName: The name of taxonomy node. creator: The name of the user creating this node.	void	Add a new taxonomy node at the given location.
removeTaxonomyNode (String repository, String workspace, String	repository: The name of the repository	void	Remove the taxonomy node located at the given absolute path.

Method	Param	Return	Description
absPath) throws RepositoryException;	workspace: The name of the workspace. absPath: The absolute path of the taxonomy node to remove.		
moveTaxonomyNode (String repository, String workspace, String srcPath, String destPath, String type) throws RepositoryException;	repository: The name of the repository. workspace: The name of the workspace. srcPath: The source path of this taxonomy. destPath: The destination path of the taxonomy. type: If type is equal to cut , the process will be cut. If type is equal to copy , the process will be copied.	void	Copy or cut the taxonomy node from the source path to the destination path. The parameter type indicates if the node must be cut or copied.
hasCategories (Node node, String taxonomyName) throws RepositoryException;	node: The node to check. taxonomyName: The name of the taxonomy.	boolean	Return true if the given node has categories in the given taxonomy.
hasCategories ((Node node, String taxonomyName, boolean system) throws RepositoryException;	node: The node to check. taxonomyName: The name of the taxonomy. system: Check system provider or not.	boolean	Return true if the given node has categories in the given taxonomy.
getCategories (Node node, String taxonomyName) throws RepositoryException;	node: The node for which we seek the categories.	List<Node>	Return all the paths of the categories (relative to the root node of the given taxonomy) which have been associated to the

Method	Param	Return	Description
	taxonomyName: The name of the taxonomy.		given node for the given taxonomy.
getCategories (Node node, String taxonomyName, boolean system) throws RepositoryException;	node: The node for which we seek the categories. taxonomyName: The name of the taxonomy. system: Check system provider or not.	List<Node>	Return all the paths of the categories(relative to the root node of the given taxonomy) which have been associated to the given node for the given taxonomy.
getAllCategories (Node node) throws RepositoryException;	node: The node for which we seek the categories	List<Node>	Return all the paths of the categories which have been associated to the given node.
getAllCategories (Node node, boolean system) throws RepositoryException;	node: The node for which we seek the categories system: Check system provider or not.	List<Node>	Return all the paths of the categories which have been associated to the given node.
removeCategory (Node node, String taxonomyName, String categoryPath) throws RepositoryException;	node: The node from which the category is removed. taxonomyName: The name of the taxonomy. categoryPath: The path of the category relative to the root node of the given taxonomy	void	Remove a category to the given node.
removeCategory (Node node, String taxonomyName, String categoryPath, boolean system) throws RepositoryException;	node: The node from which the category is removed. taxonomyName: The name of the taxonomy. categoryPath: The path of the category relative to the root node of the given	void	Remove a category to the given node.

Method	Param	Return	Description
	taxonomy. system: Check system provider or not.		
addCategories (Node node, String taxonomyName, String[] categoryPaths) throws RepositoryException;	node: The node to which we add the categories. taxonomyName: The name of the taxonomy. categoryPaths: An array of category paths relative to the given taxonomy.	void	Add several categories to the given node.
addCategories (Node node, String taxonomyName, String[] categoryPaths, boolean system) throws RepositoryException;	node: The node to which we add the categories. taxonomyName: The name of the taxonomy. categoryPaths: An array of category paths relative to the given taxonomy. system: Check system provider or not.	void	Add several categories to the given node.
addCategory (Node node, String taxonomyName, String categoryPath) throws RepositoryException;	node: The node to which we add the category. taxonomyName: The name of the taxonomy. categoryPath: The path of the category relative to the given taxonomy.	void	Add a new category path to the given node.
addCategory (Node node, String taxonomyName, String categoryPath, boolean system) throws RepositoryException;	node: The node to which we add the category. taxonomyName: The name of the taxonomy.	void	Add a new category path to the given node.

Method	Param	Return	Description
	categoryPath: The path of the category relative to the given taxonomy. system: Check system provider or not.		
getTaxonomyTreeDefaultUserPermission();		Map<String, String[]>	Get the default permission for the user in taxonomy tree.
addTaxonomyPlugin(Component plugin);	plugin: The plugin to add.	void	Add a new taxonomy plugin to the service.
init(String repository) throws Exception;	repository: The name of repository.	void	Initialize all taxonomy plugins that have been already configured in .xml files.
getCategoryNameLength()	N/A	string	Get the limited length of the category name.

4.5.2. LinkManager

Supply API to work with the linked node or the link included in a node.

Package *org.exoplatform.services.cms.link.LinkManager*

Method	Param	Return	Description
createLink (Node parent, String linkType, Node target) throws RepositoryException;	parent: The parent node of the link. linkType: The primary node type of the link must be a sub-type of <code>exo:symlink</code> , the default value is "exo:symlink" target: The target of the link.	Node	Create a new link that is added to the parent node and return the link.
createLink (Node parent, Node target) throws RepositoryException;	parent: The parent node of the link to create. target: The target of the link.	Node	Create a new node of type <code>exo:symlink</code> , then add it to the parent node and return the link node.

Method	Param	Return	Description
createLink (Node parent, String linkType, Node target, String linkName) throws RepositoryException;	parent: The parent node of the link. linkType: The primary node type of the link must be a sub-type of <code>exo:symlink</code> , the default value is <code>exo:symlink</code> . target: The target of the link. linkName: The name of the link.	Node	Create a new link that is added to the parent node and return the link.
updateLink (Node link, Node target) throws RepositoryException;	link: The link node to update. target: The new target of the link.	Node	Update the target node of the given link.
getTarget (Node link, boolean system) throws ItemNotFoundException, RepositoryException;	link: The node of type <code>exo:symlink</code> . system: Indicate whether the target node must be retrieved using a session system or user session in case we cannot use the same session as the node link because the target and the link are not in the same workspace.	Node	Get the target node of the given link.
getTarget (Node link) throws ItemNotFoundException, RepositoryException;	link: The node of type <code>exo:symlink</code> .	Node	Get the target node of the given link using the user.
isTargetReachable (Node link) throws RepositoryException;	link: The node of type <code>exo:symlink</code> .	boolean	Check if the target node of the given link can be reached using the user session.
isTargetReachable (Node link, boolean system) throws	link: The node of type <code>exo:symlink</code> .	boolean	Check if the target node of the given link can be reached using the user

Method	Param	Return	Description
<code>RepositoryException;</code>	<code>system</code>		<code>session.</code>
isLink (Item item) throws <code>RepositoryException;</code>	item: The item to test.	boolean	Indicate whether the given item is a link. @return <code>true</code>: if the node is a link, <code>false</code> otherwise.
getTargetPrimaryNodeType (Node link) throws <code>RepositoryException;</code>	link: The node of type <code>exo:symlink</code> .	string	Return the primary node type of the target.
getAllLinks (Node targetNode, String linkType, String repoName) throws <code>Exception</code>	targetNode: The target node to get links. linkType: The type of link to get. repoName: The name of the repository.	List<Node> - the list of link of the target node with given type.	Return all links of the given node.

4.5.3. PublicationManager

PublicationService is to manage the publication.

Method	Param	Return	Description
addLifecycle (ComponentPlugin plugin)	plugin	void	Add plugins context for the publication service.
getLifecycle (String name)	name: The name of the wanted lifecycle.	Lifecycle	Get a specific lifecycle with the given name.
getLifecycles ()	N/A	List<Lifecycle>	Get all the lifecycles which were added to service instances.
getContext (String name)	name	Context	Get a specific context with the given names.
getContexts ()	N/A	List<Context>	Get all the contexts which were added to service instances.
getContents (String fromstate, String tostate, String date, String user, String lang , String ...)	fromstate/tostate: The current range state of node.	List<Node>	Get all the nodes.

Method	Param	Return	Description
workspace) throws Exception;	user: The last user of node. lang: The node's language. workspace: The Workspace of the node's location.		
getLifecyclesFromUser (String remoteUser, String state);	remoteUser: The current user of publication service. state: The current state of the node.	List<Lifecycle>	Get all the Lifecycle of a specific user.

4.5.4. WCMComposer

This class is used to get contents inside the WCM product. You should not access contents directly from the JCR on the front side.

In general, this service stands between publication and cache.

Package *org.exoplatform.services.wcm.publication.WCMComposer*

Method	Param	Return	Description
getContent (String repository, String workspace, String nodeIdentifier, HashMap<String, String> filters, SessionProvider sessionProvider)throws Exception;	repository workspace nodeIdentifier filters sessionProvider: The session provider.	Node	Return contents at the specified path based on filters.
getContents (String repository, String workspace, String path, HashMap<String, String> filters, SessionProvider sessionProvider)	repository workspace	List<Node>	Return contents at the specified path based on filters.

Method	Param	Return	Description
sessionProvider)throws Exception;	path filters sessionProvider: The session provider.		
updateContent (String repository, String workspace, String nodeIdentifier, HashMap<String, String> filters)throws Exception;	epository workspace path filters	boolean	Update content.
getAllowedStates String mode)throws Exception;	mode	List<String>	Return allowed states for a specified mode.
cleanTemplates ()throws Exception;	N/A	void	Initialize the template hashmap.
isCached ()throws Exception;	N/A	boolean	Check isCache or not.
updateTemplatesSQLFilter ()throws Exception;		String	Update all document nodetypes and write a query cause. It returns a part of the query that allows to search all document nodes and taxonomy links. Return null if there is any exception.

4.5.5. NewFolksonomy

Package *org.exoplatform.services.cms.folksonomy.NewFolksonomyService*;

Method	Return	Prototype	Description
addPrivateTag (String[] tagsName, Node documentNode, String repository, String workspace, String userName) throws Exception ;	tagNames: The array of tag name as the children of tree. documentNode: Tag this node by creating a	void	Add a private tag to a document. A folksonomy link will be created in a tag node.

Method	Return	Prototype	Description
	folksonomy link to the node in the tag. repository: The repository name. workspace: The workspace name. userName: The username.		
addGroupsTag (String[] tagsName, Node documentNode,String repository, String workspace, String[] roles) throws Exception ;	tagNames: The array of tag name as the children of tree. documentNode: Tag this node by creating a folksonomy link to the node in tag. repository: The repository name. workspace: The workspace name. roles: The user roles.	void	Add a group tag to a document. A folksonomy link will be created in a tag node.
addPublicTag (String treePath, String[] tagsName, Node documentNode, String repository, String workspace) throws Exception ;	treePath: The path of folksonomy tree. tagNames: The array of the tag name as the children of tree. documentNode: Tag this node by creating a folksonomy link to the node in the tag. repository: The repository name.	void	Add a public tag to a document. A folksonomy link will be created in a tag node.

Method	Return	Prototype	Description
	workspace: The workspace name.		
addSiteTag (String siteName, String[] tagsName, Node node, String repository, String workspace) throws Exception ;	siteName: The portal name. treePath: The path of folksonomy tree. tagNames: The array of the tag name as the children of tree. documentNode: Tag this node by creating a folksonomy link to the node in tag. repository: The repository name. workspace: The workspace name.	void	Add a site tag to a document. A folksonomy link will be created in a tag node.
getAllPrivateTags (String userName, String repository, String workspace) throws Exception ;	userName: The user name. repository: The repository name. workspace: The workspace name.	List<Node>	Get all private tags.
getAllPublicTags (String treePath, String repository, String workspace) throws Exception ;	treePath: The folksonomy tree path. repository: The repository name. workspace: The workspace name.	List<Node>	Get all public tags.
getAllGroupTags (String[] roles, String repository, String workspace) throws	roles: The roles of user.	List<Node>	Get all tags by groups.

Method	Return	Prototype	Description
Exception ;	repository: The repository name. workspace: The workspace name.		
getAllGroupTags (String role, String repository, String workspace) throws Exception ;	role: The role of user. repository: The repository name. workspace: The workspace name.	List<Node>	Get all tags by a group.
getAllSiteTags (String siteName, String repository, String workspace) throws Exception ;	siteName: The portal name. treePath: Folksonomy tree path. repository: The repository name. workspace: The workspace name.	List<Node>	Get all tags of Site.
getAllDocumentsByTag (String tagPath, String repository, String workspace, SessionProvider sessionProvider) throws Exception ;	treeName: The name of folksonomy tree. tagName: The name of a tag. repository: The repository name.	List<Node>	Get all documents which are stored in a tag and return a list of documents in a tag.
getTagStyle (String tagPath, String repository, String workspace) throws Exception ;	tagPath workspace: The workspace name. repository: The repository name.	string	Get HTML_STYLE_PROP property in styleName node in the repository.

Method	Return	Prototype	Description
addTagStyle (String styleName, String tagRange, String htmlStyle, String repository, String workspace) throws Exception ;	styleName: The style name. tagRate: The range of tag numbers. htmlStyle: The tag style. repository: The repository name. workspace: The workspace name.	void	Update the properties TAG_RATE_PROP and HTML_STYLE_PROP, following the values tagRate, htmlStyle for a node in tagPath in repository.
updateTagStyle (String styleName, String tagRange, String htmlStyle, String repository, String workspace) throws Exception ;	styleName: The style name. tagRate: The range of tag numbers. htmlStyle: The tag style. repository: The repository name. workspace: The workspace name.	void	Update the properties TAG_RATE_PROP and HTML_STYLE_PROP, following the value tagRate, htmlStyle for a node in tagPath in repository.
getAllTagStyle (String repository, String workspace) throws Exception ;	repository: The repository name workspace: The workspace name.	List<Node>	Get all tag style bases of a folksonomy tree.
init (String repository) throws Exception ;	repository: The repository name.	void	Initialize all TagStylePlugin with session in repository name.
removeTagOfDocument (String tagPath, Node document, String repository, String workspace) throws Exception;	treeName: The name of a folksonomy tree. tagName: The name of a	void	Remove a tag of a given document.

Method	Return	Prototype	Description
	tag. document: The document which is added a link to tagName. repository: The repository name.		
removeTag (String tagPath, String repository, String workspace) throws Exception;	tagPath: The path of the tag. repository: The repository name. workspace: The workspace name.	void	Remove a tag.
modifyTagName (String tagPath, String newTagName, String repository, String workspace) throws Exception;	tagPath: The name of the tag. newTagName: The new tag name. repository: The repository name. workspace: The workspace name.	Node	Modify the tag name.
getLinkedTagsOfDocument (Node documentNode, String repository, String workspace) throws Exception;	documentNode: The document node. repository workspace	List<Node>	Get all tags linked to a given document.
getLinkedTagsOfDocumentByScope (int scope, String value, Node documentNode, String repository, String workspace) throws Exception;	documentNode: The document node. repository: The repository name.	List<Node>	Get all tags linked to a given document by scope.

Method	Return	Prototype	Description
	workspace: The workspace name.		
removeTagsOfNodeRecursively (Node node, String repository, String workspace, String username, String groups) throws Exception;	repository workspace	void	Remove all tags linked to the child nodes of a given node.
addTagPermission (String usersOrGroups);	usersOrGroups	void	Add given users or groups to tagPermissionList.
removeTagPermission (String usersOrGroups);	usersOrGroups	void	Remove given users or groups from tagPermissionList.
getTagPermissionList ();	N/A	List<String>	Return tagPermissionList.
canEditTag (int scope, List<String> memberships);	scope memberships	boolean	Set the permission to edit a tag for a user.
getAllTagNames (String repository, String workspace, int scope, String value) throws Exception;	repository workspace scope: The scope of tags. value: The value, according to scope, can be understood differently.	List<String>	Get all tag names which start within a given scope.

4.5.6. ApplicationTemplateManager

This class is used to manage dynamic groovy templates for WCM-based products.

Package org.exoplatform.services.cms.views.ApplicationTemplateManager;

Method	Param	Return	Description
addPlugin (PortletTemplatePlugin portletTemplatePlugin) throws Exception	PortletTemplatePlugin	void	Add the plugin..
getAllManagedPortletNames (String repository)	repository	List<String>	Retrieve all the portlet

Method	Param	Return	Description
repository) throws Exception			names that have dynamic groovy templates managed by service.
getTemplatesByApplication (String repository, String portletName, SessionProvider provider)throws Exception;	repository portletName provider: The provider.	List<String>	Retrieve the templates node by application.
getTemplatesByCategory (String repository, String portletName, String category, SessionProvider sessionProvider) throws Exception;	repository portletName category sessionProvider	List<String>	Retrieve the templates node by category.
getTemplateByName (String repository, String portletName, String category, String templateName, SessionProvider sessionProvider)throws Exception;	repository portletName category templateName sessionProvider	node	Retrieve the template by name.
getTemplateByPath (String repository, String templatePath, SessionProvider sessionProvider)throws Exception ;	repository templatePath sessionProvider	node	Get the template by path.
addTemplate (node portletTemplateHome, PortletTemplateConfig config)throws Exception;	portletTemplateHome config	void	Add the template.
removeTemplate (String repository, String portletName, String catgory, String	repository portletName	void	Remove the template.

Method	Param	Return	Description
templateName, SessionProvider sessionProvider)throws Exception;	catgory templateName sessionProvider		

4.5.7. NodeFinder

NodeFinder is used to find a node with a given path. If the path to the node contains sub-paths to `exo:symlink` nodes, find the real link node.

Method	Param	Return	Description
getNode (Node ancestorNode, String relativePath) throws PathNotFoundException, RepositoryException;	ancestorNode: The ancestor of the node to retrieve from which we start. relativePath: The relative path of the node to retrieve.	node Return the node at relPath related to the ancestor node.	
getNode (Node ancestorNode, String relativePath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	ancestorNode: The ancestor of the node to retrieve from which we start. relativePath: The relative path of the node to retrieve. giveTarget: Indicate if the target must be returned in case the item is a link.	node Return the node at relPath related to the ancestor node. If the node is a link and giveTarget has been set to <code><code>true</code></code> , the target node will be returned.	
getItem (String repository, String workspace, String absPath) throws PathNotFoundException, RepositoryException;	repository: The repository name. workspace: The workspace name. absPath: An absolute	item Return the item at the specified absolute path.	

Method	Param	Return	Description
	path.		
getItemSys (String repository, String workspace, String absPath, boolean system) throws PathNotFoundException, RepositoryException;	repository: The name of repository workspace: The workspace name. absPath: An absolute path.	item Return the item at the specified absolute path.	
getItem (String repository, String workspace, String absPath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	repository: The repository name. workspace: The workspace name. absPath: An absolute path. giveTarget: Indicate if the target must be returned in case the item is a link.	item Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code> , the target node will be returned.	
getItemGiveTargetSys (String repository, String workspace, String absPath, boolean giveTarget, boolean system) throws PathNotFoundException, RepositoryException;	repository: The repository name. workspace: The workspace name. absPath: An absolute path. giveTarget: Indicate if the target must be returned in case the item is a link. system: The system provider.	Item	Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code> , the target node will be returned.
getItem (Session session, String absPath) throws PathNotFoundException,	session: The session is used to get the item.	item Return the item at the specified absolute path.	

Method	Param	Return	Description
RepositoryException;	absPath: An absolute path.		
getItem (Session session, String absPath, boolean giveTarget) throws PathNotFoundException, RepositoryException;	session: The session is used to get the item. absPath: An absolute path. giveTarget: Indicate if the target must be returned in case the item is a link.	item Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code>, the target node will be returned.	
getItemTarget (Session session, String absPath, boolean giveTarget, boolean system) throws PathNotFoundException, RepositoryException;	session: The session is used to get the item. absPath: An absolute path. giveTarget: Indicate if the target must be returned in case the item is a link. system: The system provider	item Return the item at the specified absolute path. If the item is a link and giveTarget has been set to <code>true</code>, the target node will be returned.	
itemExists (Session session, String absPath) throws RepositoryException;	session: The session is used to get the item. absPath: An absolute path.	booleanReturn <code>true</code> if an item exists at absPath; otherwise returns <code>false</code>. Also returns <code>false</code> if the specified absPath is malformed.	

4.5.8. JodConverter

JodConverter is used to convert documents into different office formats.

Package *org.exoplatform.services.cms.jodconverter.JodConverterService*

Method	Param	Return	Description
convert (InputStream input, String formatInput, OutputStream out, String formatOutput) throws Exception;	input formatInput out formatOutput	void	Convert InputStream in the formatInput format to OutputStream with the formatOutput format.

4.5.9. SiteSearchService

SiteSearchService is used in the Search portlet that allows users to find all information matching with your given keyword.

Method	Param	Return	Description
addExcludeIncludeDataTypePlugin (ExcludeIncludeDataTypePlugin plugin)			Filter mimetypes data in the search results.
searchSiteContents (SessionProvider sessionProvider, QueryCriteria queryCriteria, int pageSize, boolean isSearchContent) throws Exception;	queryCriteria: The query criteria for SiteSearchService. Basing on search criteria, SiteSearchService can easily create the query statement to search. sessionProvider: The session provider. pageSize: The number of search results on a page.	AbstractPageList<Result>	Find all child nodes whose contents match with the given keyword. These nodes will be put in the list of search results.

4.5.10. SEOService

SEOService supplies APIs to manage SEO data of a page or a content. This service includes some major functions which enables you to add, store, get or remove the metadata of a page or a content.

Method	Param	Return	Description
storePageMetadata (PageMetadataModel metaModel, String portalName, boolean onContent) throws Exception	PageMetadataModel metaModel: The metadata of a page/content stored. portalName: The name of portal. onContent: Indicate whether the current page is the content page or the portal page.	void	Store the metadata of a page/content.
getMetadata (ArrayList<String> params, String pageReference) throws Exception	params: The parameters list of a content page. pageReference: The reference of the page.	PageMetadataModel	Return the metadata of a portal page or a content page.
getPageMetadata (String pageReference) throws Exception	pageReference: The reference of the page.	PageMetadataModel	Return the metadata of a portal page.
getContentMetadata (ArrayList<String> params) throws Exception	params: The parameters list of a content page.	PageMetadataModel	Return the metadata of a content page.
removePageMetadata (PageMetadataModel metaModel, String portalName, boolean onContent) throws Exception	PageMetadataModel metaModel: The metadata of a page/content stored. portalName: The name of portal. onContent: Indicate whether the current page is the content page or the portal page.	void	Remove the metadata of a page.
getContentNode (String contentPath) throws Exception	contentPath: The content path.	Node	Return the content node by the content path.
getHash (String uri) throws Exception	uri: The page reference of the UUID of a node.	string	Create a key from the page reference or the UUID of the node.
getSitemap (String portalName) throws Exception	portalName: The portal name.	string	Return a sitemap's content of a specific portal.
getRobots (String portalName)	portalName: The portal	string	Return Robots' content of

Method	Param	Return	Description
portalName) throws Exception	name.		a specific portal.
getRobotsIndexOptions() throws Exception	N/A	List<String>	Return a list of options (INDEX and NOINDEX) for robots to index.
getRobotsFollowOptions() throws Exception	N/A	List<String>	Return a list of options (FOLLOW and NOFOLLOW) for robots to follow.
getFrequencyOptions() throws Exception	N/A	List<String>	Return a list of options for frequency.

4.6. FAQ

4.6.1. How to deploy a workflow?

The *addPlugin()* function of *WorkflowServiceContainer* service is used to register a Business Process when a workflow is implemented. Thus, if you want to use a workflow, you are required to configure the workflow service to invoke the *addPlugin()* function by adding the *external-component-plugins* element to the configuration file.

You have to set values for the name and location of the workflow which you want to use. There are two ways to configure the location of the workflow.

4.6.1.1. Deploy a workflow inside a .war file

You can use "war:(FOLDER_PATH)" to configure which .jar files contain your workflow processes inside the .war file.

```
<external-component-plugins>
  <target-component>org.exoplatform.services.workflow.WorkflowServiceContainer</target-component>
  <component-plugin>
    <name>deploy.predefined.processes</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.workflow.PredefinedProcessesPlugin</type>
    <init-params>
      <object-param>
        <name>predefined.processes</name>
        <description>load of default business processes</description>
        <object type="org.exoplatform.services.workflow.ProcessesConfig">
          <field name="processLocation">
            <string>war:/conf/bp</string>
          </field>
          <field name="predefinedProcess">
            <collection type="java.util.HashSet">
              <value>
                <string>/exo-ecms-ext-workflow-bp-jbpm-content-2.1.1.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-jbpm-payraise-2.1.1.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-jbpm-holiday-2.1.1.jar</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

```

        </value>
      </collection>
    </field>
  </object>
</object-param>
</init-params>
</component-plugin>
</external-component-plugins>

```

4.6.1.2. Deploy a workflow inside a .jar file

You can use *classpath:* to configure which .jar files contain your workflow processes inside the .jar file.

```

<external-component-plugins>
  <target-component>org.exoplatform.services.workflow.WorkflowServiceContainer</target-component>
  <component-plugin>
    <name>deploy.predefined.processes</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.workflow.PredefinedProcessesPlugin</type>
    <init-params>
      <object-param>
        <name>predefined.processes</name>
        <description>load of default business processes</description>
        <object type="org.exoplatform.services.workflow.ProcessesConfig">
          <field name="processLocation">
            <string>classpath:</string>
          </field>
          <field name="predefinedProcess">
            <collection type="java.util.HashSet">
              <value>
                <string>/exo-ecms-ext-workflow-bp-jbpm-content-myworkflow.jar</string>
              </value>
            </collection>
          </field>
        </object>
      </object-param>
    </init-params>
  </component-plugin>
</external-component-plugins>

```

The notification message is displayed when you deploy a workflow on Jboss. If you use *classpath:* to register, you must put your workflow in the .jar files inside the *gatein.ear/lib* folder (instead of the *lib* folder) to make it work.