Reference Guide / Content Functions

Table of Contents

About Content Package

All package actions in 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**

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

  - Make WCM extension EAR

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

  - Make WCM demo sites EAR

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

  - Make workflow EAR

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

Portlet Applications

This chapter introduces you to a list of portlets included in Content, and their details (packaging, portlet class name, available preferences and sample configurations):

- **Content Detail**

  Allow viewing details of a specific content.
- **Content List**

  Show a list of content which already exist in the system.
- **Search**

  Allow doing a search with any string.
- **Sites Explorer**

  Manage all documents in different drives. With this portlet, you 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.
- **Administration**

  Manage the main Content functions, including categories and tags, content presentation, types of content and advanced configuration.
- **Fast Content Creator**

  Consist of two modes: **Standard Content Creator** and **Basic Content Creator**. This portlet allows users to quickly create contents without accessing the Sites Explorer portlet.
- **Form Builder**

  Allow users to create node types and document templates for node types.
- **Authoring**

  Allow users to manage contents in draft and ones which need to be approved or published.
- **Newsletter**

  Help users quickly get the updated newsletter from a site.
- **SEO portlet**

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

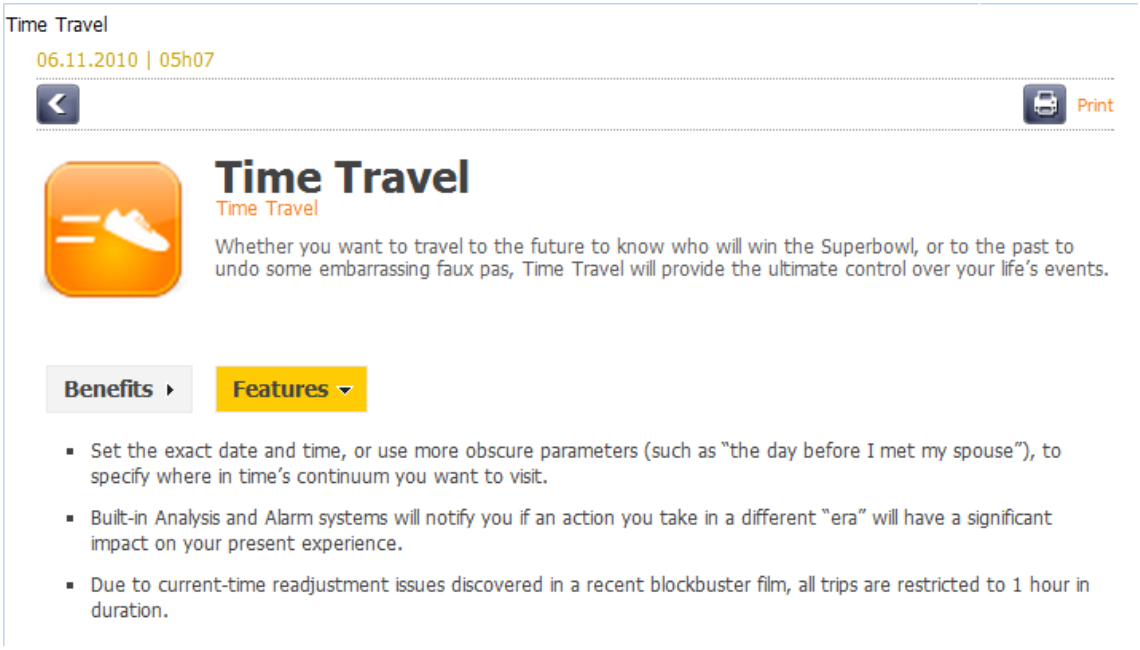
These portlet applications are packaged as Web application archives (WARs).

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*.

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 Content:



- **Packaging**: This portlet is packaged in the *presentation.war* file.
- **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 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`	

Preference	Type	Value	Description
			Define if the portlet will use the cache shared between users to display content. If you want the content displayed in CLV to be got from one cache, set the value to `true`. In most cases, you should not set **sharedCache** to `false` as it reduces the overall performance. See Content Visibility.

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

Note

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

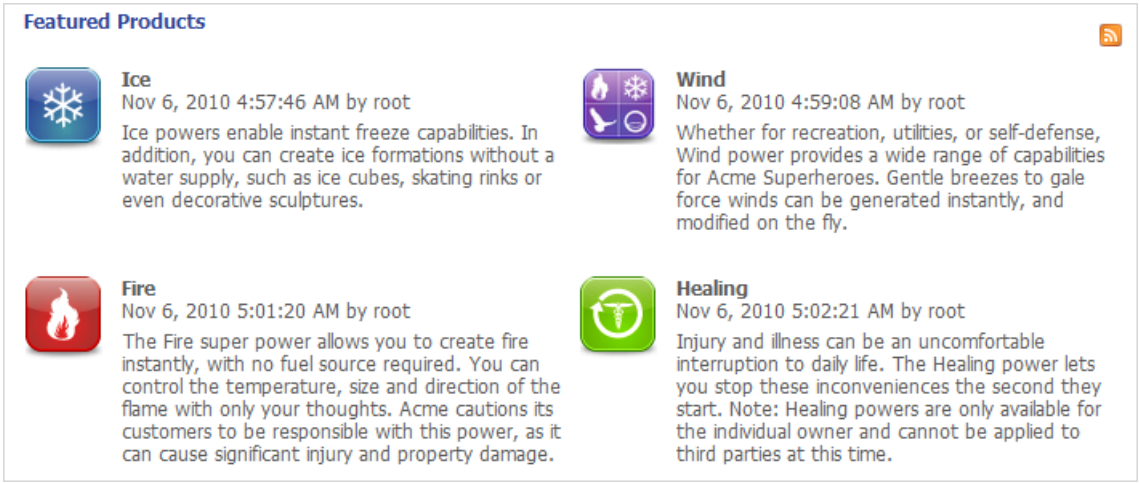
See also

- Content List
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Form Builder
- Authoring
- Newsletter
- SEO portlet

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 Content:



- **Packaging**: This porlet is packaged in the *presentation.war* file.
- **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`	

Preference	Type	Value	Description
			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 *folderPath* parameter (true value) or the value given in the *header* parameter above.
formViewTemplatePath	`String`	`/exo:ecm/views/templates/content-list-viewer/list/UIContentListPresentationDefault.gtmpl`	The path to the template used to display the contents in this portlet.
paginatorTemplatePath	`String`	`/exo:ecm/views/templates/content-list-viewer/paginators/UIPaginatorDefault.gtmpl`	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.

Preference	Type	Value	Description
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 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 use the cache shared between users to display content. If you want the content displayed in SCV to be got from one cache, set the value to `true`. In most cases, you should not

Preference	Type	Value	Description
			set **sharedCache** to `false` as it reduces the overall performance. See Content Visibility.

- **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>
```

See also

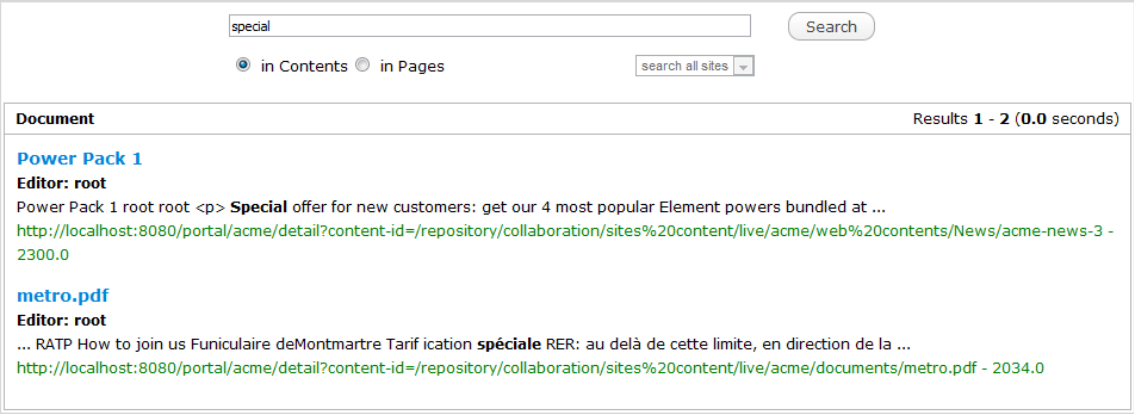
- Content Detail
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Form Builder
- Authoring
- Newsletter

- SEO portlet

1.3. Search

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

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



- **Packaging**: This portlet is packaged in the *searches.war* file.
- **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/ WCM Advance Search/ search-form/ UIDefaultSearchForm.gtmpl*	The path to the search form template.
searchResultTemplatePath	`string`	*/exo:ecm/views/templates/ WCM Advance Search/ search-result/ UIDefaultSearchResult.gtmpl*	The path to the search result template.
searchPaginatorTemplatePa	`string`	*/exo:ecm/views/templates/ WCM Advance Search/ search-paginator/ UIDefaultSearchPaginator.gtmpl*	The path to the search paginator template.
searchPageLayoutTemplatel	`string`	*/exo:ecm/views/templates/ WCM Advance Search/ search-page-layout/ UISearchPageLayoutDefault.gtmpl*	The path to the search page template.
itemsPerPage	`Integer`	5	The number of items for each page.
showQuickEditButton	`boolean`	`true`	

Preference	Type	Value	Description
			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>
```

See also

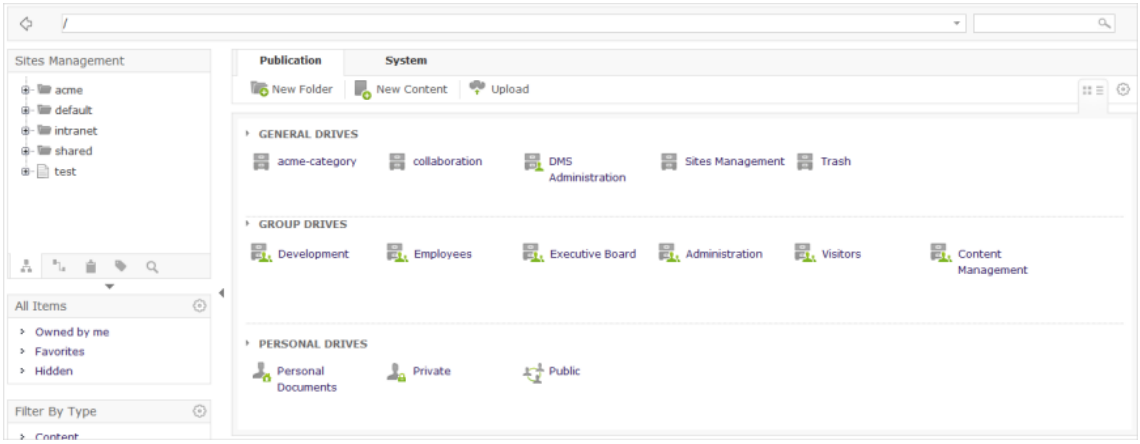
- Content Detail
- Content List
- Sites Explorer
- Administration
- Fast Content Creator

- Form Builder
- Authoring
- Newsletter
- SEO portlet

1.4. Sites Explorer

The **Sites 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 **Sites Explorer** portlet used in Content:



- **Packaging**: The porlet is packaged in the *ecmexplorer.war* file.
- **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 Sites 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 driveName 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

Preference	Type	Value	Description
			specify the value, the default value will be nt:unstructured, nt:folder.
categoryMandatoryWhenFile	`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 Sites Explorer. By default, the "selection" option is configured. Besides "selection", there are four other ways to configure the Sites 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>
```

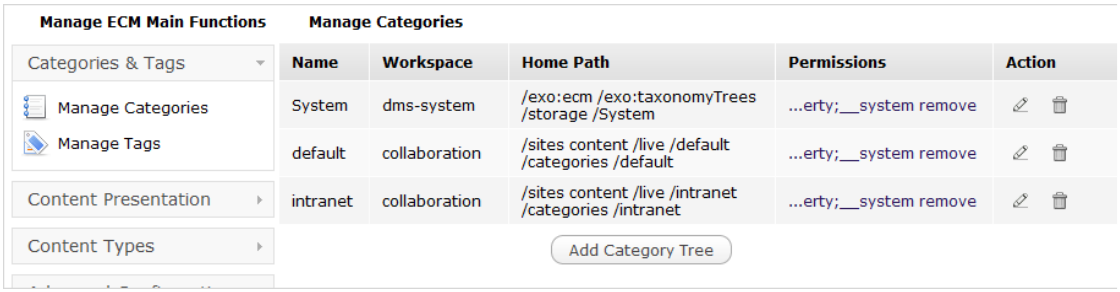
See also

- Content Detail
- Content List
- Search
- Administration
- Fast Content Creator
- Form Builder
- Authoring
- Newsletter
- SEO portlet

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 Content:



- **Packaging**: This portlet is packaged in the *ecmadmin.war* file.
- **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>
```

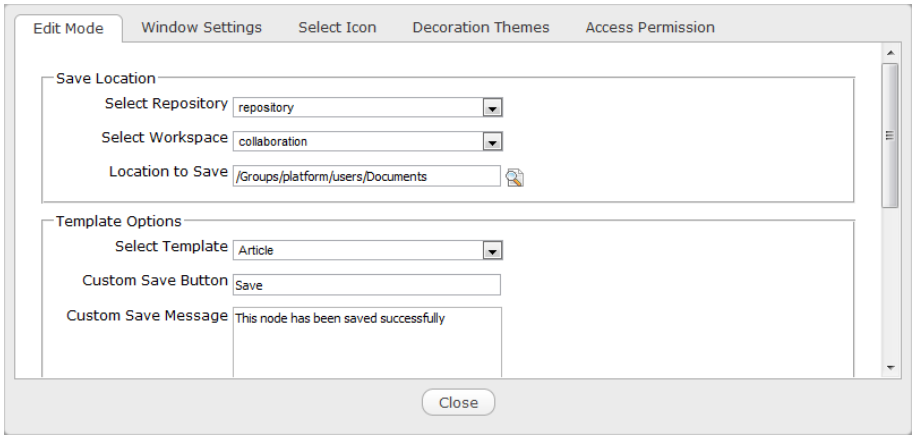
See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Fast Content Creator
- Form Builder
- Authoring
- Newsletter
- SEO portlet

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 Sites Explorer portlet.

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



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

- **Packaging**: This portlet is packaged in the *formgenerator.war* file.
- **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.
repository	`string`	`repository`	The name of the current repository.
workspace	`string`	`collaboration`	The workspace where the content is stored.

Preference	Type	Value	Description
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>
```

See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Administration
- Form Builder
- Authoring
- Newsletter
- SEO portlet

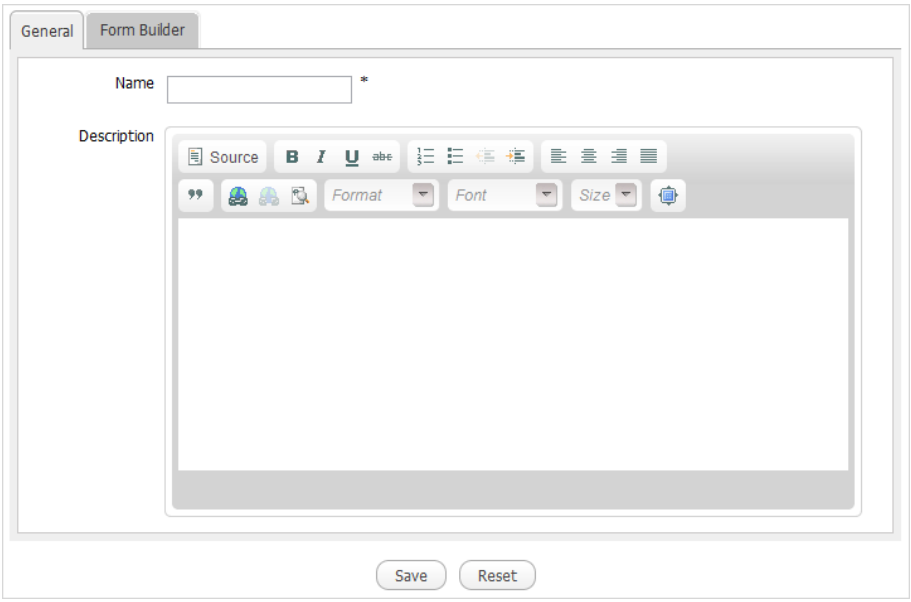
1.7. Form Builder

Note

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 Content:



- **Packaging**: This porlet is packaged in the *formgenerator.war* file.

- **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>
```

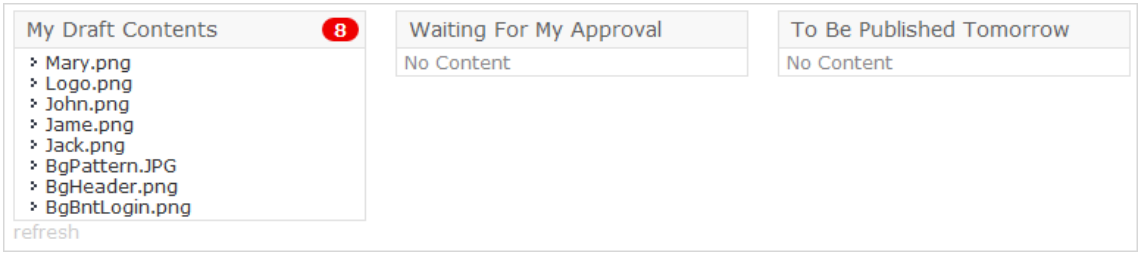
See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Authoring
- Newsletter
- SEO portlet

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 Content:



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

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

Preference	Type	Value	Description
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>
```

See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Form Builder
- Newsletter
- SEO portlet

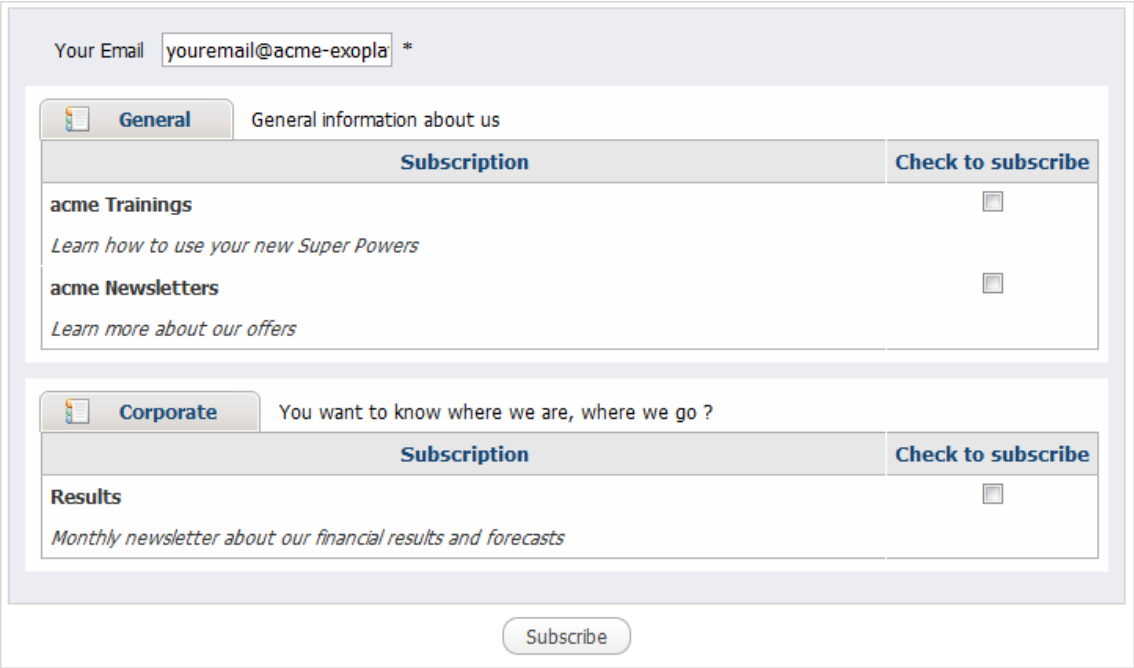
1.9. Newsletter

Note

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 Content:



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

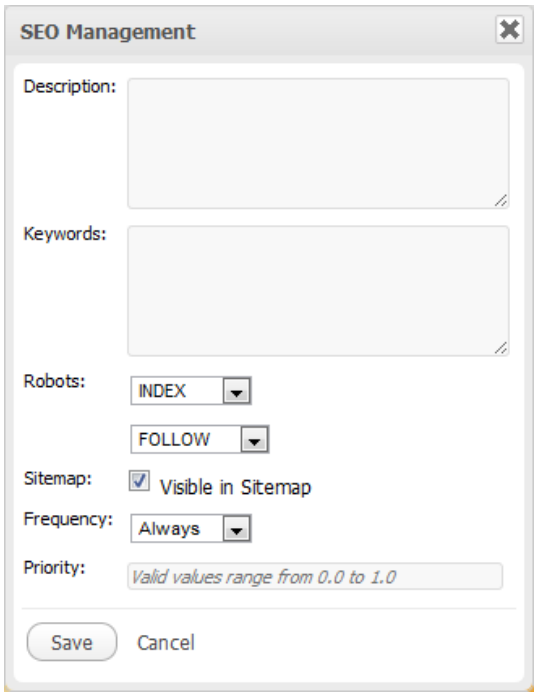
See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Form Builder
- Authoring
- SEO portlet

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 Content:



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

See also

- Content Detail
- Content List
- Search
- Sites Explorer
- Administration
- Fast Content Creator
- Form Builder
- Authoring
- Newsletter

CMIS

This chapter will help you understand what CMIS is, how it is extended in WCM, and its features via the following topics:

- **Overview**

  Overall introduction to CMIS, xCMIS and eXo CMIS.
- **CMIS specification**

  Introduction to the CMIS specification and functions of a Web services interface which is provided by the CMIS specification.
- **xCMIS project**

  Introduction to the xCMIS project and its benefits.
- **CMIS features**

  Introduction to the CMIS features, including how to integrate with eXo WCM, CMIS Domain Model and CMIS Services.
- **Service JARs**

  Additional information about a list of JARs used in eXo CMIS.

2.1. Overview

eXo Platform provides CMIS support using the xCMIS project and the WCM Storage provider.

About CMIS

The CMIS standard aims at defining a common content management web services interface that can be applied in content repositories and bring about the interoperability across repositories. The formal specification of CMIS standard is approved by the Organization for the Advancement of Structured Information Standards (OASIS) technical committee, who drives the development, convergence and adoption of global information society. With CMIS, enterprises now can deploy systems independently, and create specialized applications running over a variety of content management systems.

To see the advantages of content interoperability and the significance of CMIS as a whole, it is necessary to learn about mutual targets which caused the appearance of specification first.

- Content integration: With CMIS, integrating content among various repositories, even those created by different vendors in a single application, becomes faster, simpler and more effective. CMIS makes it possible for customers to integrate content management systems into their key business processes across business departments and vendor implementations.
- Access unification: CMIS enables different applications and manufacturers to be connected to a CMIS-enabled content repository simply. With CMIS, a business application's developer can focus on the application's business logic, rather than issues related to the compatibility or content migration.

About xCMIS

The xCMIS project, which is initially contributed to the Open Source community by eXo Platform, is an Open Source implementation of the Content Management Interoperability Services (CMIS) specification. xCMIS supports all the features stated in the CMIS core definition and both REST AtomPub and Web Services (SOAP/WSDL) protocol bindings.

Tip

To learn more about xCMIS, visit:

- http://gazarenkov.blogspot.com/2010/01/xcmis1-cmis-in-nutshell.html
- http://code.google.com/p/xcmis/

About eXo CMIS

eXo CMIS is built on the top of xCMIS embedded in eXo Platform to expose the WCM drives as the CMIS repositories. The CMIS features are implemented as a set of components deployed on the eXo Container using XML files to describe the service configuration.

Note

SOAP protocol binding is not implemented in eXo CMIS.

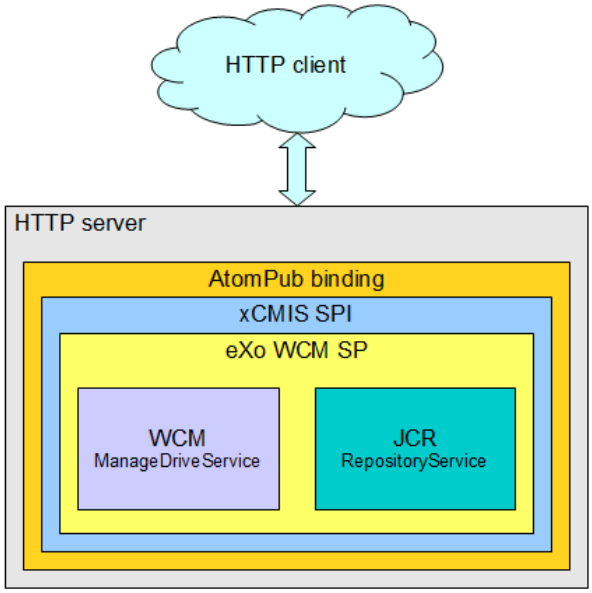


Figure: How eXo CMIS works

WCM drives exposure is implemented as a WCM storage provider to the xCMIS SPI. The storage provider uses mappings from the WCM's *ManageDriveService* to actual JCR nodes. *AtomPub* bindings makes WCM structure available via CMIS standard API.

See also

- CMIS specification
- xCMIS project
- CMIS features
- Service JARs

2.2. CMIS specification

Note

This is related to Content Management Interoperability Services (CMIS) Version 1.0 OASIS Standard 1 May 2010

http://en.wikipedia.org/wiki/Content_Management_Interoperability_Services

The CMIS interface is designed to be layered on top of existing Content Management systems and their existing programmatic interfaces. It is intended to expose all of the CM systems capabilities through the CMIS interfaces exhaustively. The CMIS specification defines the followings:

- A standard "domain model" for an ECM system - a set of core concepts included in all modern ECM systems, such as Object Types, properties, folders, documents, versions, and relationships; and a set of operations performed on those concepts, such as updating documents, or navigating via a folder hierarchy.
- The way to bind the CMIS domain model to two different web service protocols, including the Simple Object Access Protocol (SOAP) used in many ECM systems, and the Atom used in many Web 2.0 applications.

Note

The SOAP protocol is not implemented in eXo CMIS.

The CMIS specification provides a Web services interface which can:

- Work over existing repositories, enabling customers to build and leverage applications against multiple repositories.
- Decouple Web services and content from the content management repository, enabling customers to manage content independently.
- Provide common Web services and Web 2.0 interfaces to dramatically simplify the application development.
- Build the development platform and language agnostic.
- Support the composite application development and mashups by the business or IT analysts.

See also

- Overview
- xCMIS project
- CMIS features
- Service JARs

2.3. xCMIS project

xCMIS includes the client side frameworks for integrating content from different enterprise repositories, according to CMIS standard.

The project is to make joining Enterprise Content repositories simpler by offering CMIS abilities and exposing them to language-independent CMIS clients via the most convenient protocol.

xCMIS project:

- Is embedded, packaged as the J2EE Web archive (WAR) and prepared "download and go" Tomcat bundle.

- Has a live demo with the full-featured CMIS Expert client, which is accessible via xcmis.org site and with prepared "download and go" Tomcat bundle (the client is accessible as the remote gadget).
- Is embedded in eXo Platform to create the special xCMIS jcr repository and access it with any CMIS client.
- Tested with third-party CMIS clients, such as IBM CMIS Firefox Connector and CMIS Spaces Flex+AIR client. Either local repository (as described here), or can be used as a CMIS repository's endpoint URL for these, or other types of clients.

Benefits of xCMIS:

- xCMIS is an open source, server side Java CMIS implementation, enabling to expose content in the existing content repositories according to the protocols defined in the CMIS specification.
- xCMIS will give developers a way to make their content repositories "pluggable" on the server side based on the internal Storage Provider Interface and additional protocol on-demand bindings.
- xCMIS will provide (several) CMIS client frameworks for repository-application and repository-repository interactions. The programming language and supported protocol can be selected by users. For example, the reasonable choice for using web applications, gadgets, and/or mashups is JavaScript, or GWT over REST AtomPub, while for inter-repository exchange, it may be Java over Web Services like WSDL/SOAP.
- Both the server and client sides of xCMIS are easily integrated in eXo Platform 3.0 infrastructure. In particular, xCMIS exposes the eXo JCR content repository and provides a framework for building web applications and gadgets for the GateIn portal.

The xCMIS project is distributed under the LGPL license. You can download sources on Google code, or visit Community Wiki for more information.

See also

- Overview
- CMIS specification
- CMIS features
- Service JARs

2.4. CMIS features

- **Integration with eXo WCM**

  This section will show you how to integrate between eXo WCM and CMIS via **JCR namespaces and nodetypes** , **WCM drives as CMIS Repositories** , **WCM Symlinks** , **CMIS search** , and **how to modify WCM via CMIS** .
- **CMIS Domain Model**

  Necessary information about the CMIS Domain Model and some of its common entities.
- **CMIS Services**

  Introduction to the CMIS Services, including Repository, Navigation, Object, Multi-filing, Discovery, Versioning, Relationship, Policy and ACL.

See also

- Overview
- CMIS specification
- xCMIS project
- Service JARs

2.4.1. Integration with eXo WCM

eXo Web Content Management (WCM) system provides CMIS access to its content storage features:

- WCM drives
- Document files and folders
- Symlinks
- Categories

To expose WCM drives as CMIS repositories there is a special extension of *CmisRegistry*. Read the admin guide for the configuration of *org.exoplatform.ecms.xcmis.sp.jcr.exo.DriveCmisRegistry* component.

Work with CMIS is based on reference documents returned by services. Each CMIS service returns response containing links to other services describing the Document or operations on it. In most cases, a Document will be asked by its ID. Some services accepts a Document path.

Note

Notes for use cases: To access the eXo CMIS services from the client side, use the Curl tool. The CMIS AtomPub binding which is based upon the Atom (RFC4287) and Atom Publishing Protocol (RFC5023) will be used.

SOAP binding is not implemented in eXo Platform 3.x.

2.4.1.1. JCR namespaces and nodetypes

CMIS uses special JCR namespaces *cmis* and *xcmis* internally.

To expose drives content following nodetypes supported:

- nt:file nodetype for representation of cmis:documents
- nt:folder for representation of cmis:folder

Since the CMIS specification does not allow having more root types except ones described above (cmis:documents and cmis:folder), those two (nt:file and nt:folder) are mapped to CMIS types.

There are two more nodetypes which are used: cmis:policy and cmis:relationship which represent CMIS types with corresponded (see Services description for details).

Additionally, nodetypes used in WCM are mapped as follow:

- nt:unstructured + extensions as cmis:folder
- exo:taxonomy + extensions as cmis:folder

In other words only nodetypes extending nt:file, nt:folder, nt:unstructured and exo:taxonomy will be exposed correctly via CMIS API.

Warning

WCM's nodetype exo:article is not supported by eXo CMIS due to uncompliant structure to nt:file.

2.4.1.2. WCM drives as CMIS Repositories

The WCM drive is used to expose as an isolated repository via the CMIS service. Operations on the repository will reflect the drive immediately.

Tip

When working with CMIS repositories, it is important to understand that a repository reflects a WCM Drive, which is a sub-tree in JCR workspace. Two or more drives can be mapped to a same workspace or a sub-tree. As a result, changes in one repository can affect others. Refer to the WCM drives mappings to know actual location of a content you will access or change.

Use Case: Browse Drives via getRepository

- Login to the website as a user with the developer role.
- Open **Group --> Sites Explorer**, you can see the drives set of WCM, such as **Sites Management**, **DMS Administration**.
- Get the list of these WCM drives via CMIS using *Curl*, asking *getRepositories* service:

```
curl -o getrepos.xml -u root:gtn http://localhost:8080/rest/private/cmisatom/
```

The requested file (*getrepos.xml*) contains the set of Repositories in the AtomPub format. The root element represents the set of **workspaces** representing **WCM drives** related to resources, for example, for **DMS Administration**, the response will contain data like:

```
<service xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/">
<workspace>
  <atom:title type="text">DMS Administration</atom:title>
  <cmisra:repositoryInfo xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
    <cmis:repositoryId>DMS Administration</cmis:repositoryId>
    <cmis:repositoryName>DMS Administration</cmis:repositoryName>
  </cmisra:repositoryInfo>
  <collection href="http://localhost:8080/rest/private/cmisatom/DMS%20Administration/query">
    <atom:title type="text">Query</atom:title>
    <cmisra:collectionType>query</cmisra:collectionType>
  </collection>
  <collection href="http://localhost:8080/rest/private/cmisatom/DMS%20Administration/children/00exo0jcr0root0uuid0000000000000">
    <atom:title type="text">Folder Children</atom:title>
    <cmisra:collectionType>root</cmisra:collectionType>
  </collection>
  <collection href="http://localhost:8080/rest/private/cmisatom/DMS%20Administration/checkedout">
    <atom:title type="text">Checkedout collection</atom:title>
    <cmisra:collectionType>checkedout</cmisra:collectionType>
  </collection>
  <collection href="http://localhost:8080/rest/private/cmisatom/DMS%20Administration/unfiled">
    <atom:title type="text">Unfiled collection</atom:title>
    <cmisra:collectionType>unfiled</cmisra:collectionType>
  </collection>
  <collection href="http://localhost:8080/rest/private/cmisatom/DMS%20Administration/types">
    <atom:title type="text">Types Children</atom:title>
    <cmisra:collectionType>types</cmisra:collectionType>
  </collection>
  <cmisra:uritemplate>
                                                    <cmisra:template>http://localhost:8080/rest/private/cmisatom/DMS%20Administration/object/
{id}?filter={filter}&amp;includeAllowableActions={includeAllowableActions}&amp;includePolicyIds={includePolicyIds}&amp;includeRelationships={includeRelationships}&amp;includeACL={inc
    </cmisra:template>
    <cmisra:type>objectbyid</cmisra:type>
    <cmisra:mediatype>application/atom+xml;type=entry</cmisra:mediatype>
  </cmisra:uritemplate>
  <cmisra:uritemplate>
                                                        <cmisra:template>http://localhost:8080/rest/private/cmisatom/DMS%20Administration/
objectbypath?path={path}&amp;filter={filter}&amp;includeAllowableActions={includeAllowableActions}&amp;includePolicyIds={includePolicyIds}&amp;includeRelationships={includeRelations
    </cmisra:template>
    <cmisra:type>objectbypath</cmisra:type>
```

```
        <cmisra:mediatype>application/atom+xml;type=entry</cmisra:mediatype>
      </cmisra:uritemplate>
      <cmisra:uritemplate>
                                                                    <cmisra:template>http://localhost:8080/rest/private/cmisatom/DMS%20Administration/
query?q={q}&amp;searchAllVersions={searchAllVersions}&amp;maxItems={maxItems}&amp;skipCount={skipCount}&amp;includeAllowableActions={includeAllowableActions}=&amp;include
        </cmisra:template>
        <cmisra:type>query</cmisra:type>
        <cmisra:mediatype>application/atom+xml;type=feed</cmisra:mediatype>
      </cmisra:uritemplate>
      <cmisra:uritemplate>
        <cmisra:template>http://localhost:8080/rest/private/cmisatom/DMS%20Administration/typebyid/{id}
        </cmisra:template>
        <cmisra:type>typebyid</cmisra:type>
        <cmisra:mediatype>application/atom+xml;type=entry</cmisra:mediatype>
      </cmisra:uritemplate>
    </workspace>
  </service>
```

Here are the collection of services and predefined templates which can be used from the client side to request resources related to this repository. For example, to get the WCM node of the **DMS Administration** drive by path, the **objectbypath** template can be used:

```
http://localhost:8080/rest/private/cmisatom/DMS%20Administration/
objectbypath?path={path}&amp;filter={filter}&amp;includeAllowableActions={includeAllowableActions}&amp;includePolicyIds={includePolicyIds}&amp;includeRelationships={includeRelations
```

where parameters include:

- Required:
  - ID repositoryId: The identifier for the repository.
  - String path: The path to the object.
- Optional:
  - String filter
  - Boolean includeAllowableActions
  - Enum includeRelationships
  - String renditionFilter
  - Boolean includePolicyIds
  - Boolean includeACL

Note

Find full description of all specified services in the CMIS specification.

2.4.1.3. WCM Symlinks

Symlinks are used to organize the virtual access to documents in WCM, which is implemented like links in Unix/Linux/Mac OS (refer to ln command for more details).

Note

JCR exo:symlink nodetype is used for such nodetypes.

Use Case: Follow Symlinks

1. Login to the ACME website as a user with the developer role.

2. Open **Group --> Sites Explorer --> Sites Management**, go to the folder */acme/documents*.

3. Upload any file (for example *test.txt*) to */acme/documents*.

4. Add this file to the **acme/News** category. It will create a symlink to */acme/documents/test.txt* in */acme/categories/acme/News*.

5. Get content of folder */acme/categories/acme/News* via CMIS:

```
curl -o news.xml -u root:gtn http://localhost:8080/rest/private/cmisatom/Managed%20Sites/objectbypath?path=/acme/categories/acme/News
```

The requested file (*products.xml*) contains the entry with information about the folder.

- The list of properties for the object (such as node Id or type).
- Allowable actions can be performed on the document; for example, requesting the children list.
- ACL and policies information.

```
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/">
  <id>f59a3539c0a80003625790bdadf566c5</id>
  <published>2010-09-09T18:11:57.707Z</published>
  <updated>2010-09-09T18:11:57.707Z</updated>
  <summary type="text"/>
  <author>
    <name>system</name>
  </author>
  <title type="text">News</title>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites" rel="service" type="application/atomsvc+xml"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f59a3539c0a80003625790bdadf566c5" rel="self"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f59a3539c0a80003625790bdadf566c5" rel="edit"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/typebyid/exo%3Ataxonomy" rel="describedby" type="application/atom+xml; type=entry"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/allowableactions/f59a3539c0a80003625790bdadf566c5" rel="http://docs.oasis-open.org/ns/cmis/link/200908/allowableactions" type="application/cmis+xml; type=allowableActions"/>
   <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/relationships/f59a3539c0a80003625790bdadf566c5" rel="http://docs.oasis-open.org/ns/cmis/link/200908/relationships" type="application/atom+xml; type=feed"/>
   <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/policies/f59a3539c0a80003625790bdadf566c5" rel="http://docs.oasis-open.org/ns/cmis/link/200908/policies" type="application/atom+xml; type=feed"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/objacl/f59a3539c0a80003625790bdadf566c5" rel="http://docs.oasis-open.org/ns/cmis/link/200908/acl" type="application/cmisacl+xml"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/children/f59a3539c0a80003625790bdadf566c5" rel="down" type="application/atom+xml; type=feed"/>
     <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/descendants/f59a3539c0a80003625790bdadf566c5" rel="down" type="application/cmistree+xml"/>
  <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/foldertree/f59a3539c0a80003625790bdadf566c5" rel="http://docs.oasis-open.org/ns/cmis/link/200908/foldertree" type="application/atom+xml; type=feed"/>
     <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f59a3533c0a8000339af97059f243a25" rel="up" type="application/atom+xml; type=entry"/>
  <content type="text">News</content>
  <cmisra:object xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
    <cmis:properties>
                                        <cmis:propertyId displayName="cmis:allowedChildObjectTypeIds" localName="cmis:allowedChildObjectTypeIds" propertyDefinitionId="cmis:allowedChildObjectTypeIds" queryName="cmis:allowedChildObjectTypeIds"/>
      <cmis:propertyString displayName="cmis:path" localName="cmis:path" propertyDefinitionId="cmis:path" queryName="cmis:path">
        <cmis:value>/acme/categories/acme/News</cmis:value>
      </cmis:propertyString>
     <cmis:propertyId displayName="cmis:objectTypeId" localName="cmis:objectTypeId" propertyDefinitionId="cmis:objectTypeId" queryName="cmis:objectTypeId">
        <cmis:value>exo:taxonomy</cmis:value>
      </cmis:propertyId>
                          <cmis:propertyString displayName="cmis:lastModifiedBy" localName="cmis:lastModifiedBy" propertyDefinitionId="cmis:lastModifiedBy" queryName="cmis:lastModifiedBy"/>
      <cmis:propertyString displayName="cmis:name" localName="cmis:name" propertyDefinitionId="cmis:name" queryName="cmis:name">
        <cmis:value>News</cmis:value>
      </cmis:propertyString>
      <cmis:propertyString displayName="cmis:createdBy" localName="cmis:createdBy" propertyDefinitionId="cmis:createdBy" queryName="cmis:createdBy"/>
```

```
      <cmis:propertyId displayName="cmis:objectId" localName="cmis:objectId" propertyDefinitionId="cmis:objectId" queryName="cmis:objectId">
        <cmis:value>f59a3539c0a80003625790bdadf566c5</cmis:value>
      </cmis:propertyId>
                        <cmis:propertyDateTime displayName="cmis:creationDate" localName="cmis:creationDate" propertyDefinitionId="cmis:creationDate"
queryName="cmis:creationDate"/>
                         <cmis:propertyString displayName="cmis:changeToken" localName="cmis:changeToken" propertyDefinitionId="cmis:changeToken"
queryName="cmis:changeToken"/>
      <cmis:propertyId displayName="cmis:baseTypeId" localName="cmis:baseTypeId" propertyDefinitionId="cmis:baseTypeId" queryName="cmis:baseTypeId">
        <cmis:value>cmis:folder</cmis:value>
      </cmis:propertyId>
      <cmis:propertyId displayName="cmis:parentId" localName="cmis:parentId" propertyDefinitionId="cmis:parentId" queryName="cmis:parentId">
        <cmis:value>f59a3533c0a8000339af97059f243a25</cmis:value>
      </cmis:propertyId>
        <cmis:propertyDateTime displayName="cmis:lastModificationDate" localName="cmis:lastModificationDate" propertyDefinitionId="cmis:lastModificationDate"
queryName="cmis:lastModificationDate"/>
    </cmis:properties>
    <cmis:acl/>
    <cmis:exactACL>false</cmis:exactACL>
    <cmis:policyIds/>
    <cmis:rendition/>
  </cmisra:object>
</entry>
```

To get the file which has been uploaded above, use the **children** service to get the list of child nodes of */acme/documents*. Find this service URL in the response XML above:

```
curl -o childs.xml -u root:gtn http://localhost:8080/rest/private/cmisatom/Managed%20Sites/children/f59a3539c0a80003625790bdadf566c5
```

In the requested file (*childs.xml*), find the entry related to *test.txt* file uploaded via the **Sites Explorer**. Search for the "test.txt" name by title.

```
  <entry xmlns:cmisra="http://docs.oasis-open.org/ns/cmis/restatom/200908/">
    <id>f708e208c0a80003554babb97bd934ba</id>
    <published>2010-09-09T18:06:31.987Z</published>
    <updated>2010-09-09T18:06:31.987Z</updated>
    <summary type="text"/>
    <author>
      <name>system</name>
    </author>
    <title type="text">test.txt</title>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites" rel="service" type="application/atomsvc+xml"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f708e208c0a80003554babb97bd934ba" rel="self"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f708e208c0a80003554babb97bd934ba" rel="edit"/>
   <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/typebyid/cmis%3Adocument" rel="describedby" type="application/atom+xml; type=entry"/
>
          <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/allowableactions/f708e208c0a80003554babb97bd934ba" rel="http://docs.oasis-
open.org/ns/cmis/link/200908/allowableactions" type="application/cmis+xml; type=allowableActions"/>
      <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/relationships/f708e208c0a80003554babb97bd934ba" rel="http://docs.oasis-open.org/
ns/cmis/link/200908/relationships" type="application/atom+xml; type=feed"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/policies/f708e208c0a80003554babb97bd934ba" rel="http://docs.oasis-open.org/ns/cmis/
link/200908/policies" type="application/atom+xml; type=feed"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/objacl/f708e208c0a80003554babb97bd934ba" rel="http://docs.oasis-open.org/ns/cmis/
link/200908/acl" type="application/cmisacl+xml"/>
     <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/versions/f708a0f1c0a8000333e3681f99fab760" rel="version-history" type="application/
atom+xml; type=feed"/>
         <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/object/f708e208c0a80003554babb97bd934ba?returnVersion=latest" rel="current-
version" type="application/atom+xml; type=entry"/>
      <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/parents/f708e208c0a80003554babb97bd934ba" rel="up" type="application/atom+xml;
 type=feed"/>
    <link href="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/file/f708e208c0a80003554babb97bd934ba" rel="edit-media"/>
    <content src="http://localhost:8080/rest/private/cmisatom/Managed%20Sites/file/f708e208c0a80003554babb97bd934ba" type="text/plain"/>
    <cmisra:object xmlns:cmis="http://docs.oasis-open.org/ns/cmis/core/200908/">
      <cmis:properties>
        <cmis:propertyBoolean displayName="cmis:isLatestMajorVersion" localName="cmis:isLatestMajorVersion" propertyDefinitionId="cmis:isLatestMajorVersion"
 queryName="cmis:isLatestMajorVersion">
```

```
          <cmis:value>false</cmis:value>
        </cmis:propertyBoolean>
        <cmis:propertyInteger displayName="cmis:contentStreamLength" localName="cmis:contentStreamLength" propertyDefinitionId="cmis:contentStreamLength"
queryName="cmis:contentStreamLength">
          <cmis:value>38</cmis:value>
        </cmis:propertyInteger>
                    <cmis:propertyId displayName="cmis:contentStreamId" localName="cmis:contentStreamId" propertyDefinitionId="cmis:contentStreamId"
queryName="cmis:contentStreamId">
          <cmis:value>f708a0c2c0a800033bedeb35caddeed1</cmis:value>
        </cmis:propertyId>
      <cmis:propertyId displayName="cmis:objectTypeId" localName="cmis:objectTypeId" propertyDefinitionId="cmis:objectTypeId" queryName="cmis:objectTypeId">
          <cmis:value>cmis:document</cmis:value>
        </cmis:propertyId>
                                    <cmis:propertyString displayName="cmis:versionSeriesCheckedOutBy" localName="cmis:versionSeriesCheckedOutBy"
propertyDefinitionId="cmis:versionSeriesCheckedOutBy" queryName="cmis:versionSeriesCheckedOutBy"/>
                                        <cmis:propertyId displayName="cmis:versionSeriesCheckedOutId" localName="cmis:versionSeriesCheckedOutId"
propertyDefinitionId="cmis:versionSeriesCheckedOutId" queryName="cmis:versionSeriesCheckedOutId"/>
        <cmis:propertyString displayName="cmis:name" localName="cmis:name" propertyDefinitionId="cmis:name" queryName="cmis:name">
          <cmis:value>test.txt</cmis:value>
        </cmis:propertyString>
                                        <cmis:propertyString displayName="cmis:contentStreamMimeType" localName="cmis:contentStreamMimeType"
propertyDefinitionId="cmis:contentStreamMimeType" queryName="cmis:contentStreamMimeType">
          <cmis:value>text/plain</cmis:value>
        </cmis:propertyString>
                    <cmis:propertyId displayName="cmis:versionSeriesId" localName="cmis:versionSeriesId" propertyDefinitionId="cmis:versionSeriesId"
queryName="cmis:versionSeriesId">
          <cmis:value>f708a0f1c0a8000333e3681f99fab760</cmis:value>
        </cmis:propertyId>
                    <cmis:propertyDateTime displayName="cmis:creationDate" localName="cmis:creationDate" propertyDefinitionId="cmis:creationDate"
queryName="cmis:creationDate">
          <cmis:value>2010-09-09T18:06:31.987Z</cmis:value>
        </cmis:propertyDateTime>
                     <cmis:propertyString displayName="cmis:changeToken" localName="cmis:changeToken" propertyDefinitionId="cmis:changeToken"
queryName="cmis:changeToken"/>
                  <cmis:propertyBoolean displayName="cmis:isLatestVersion" localName="cmis:isLatestVersion" propertyDefinitionId="cmis:isLatestVersion"
queryName="cmis:isLatestVersion">
          <cmis:value>true</cmis:value>
        </cmis:propertyBoolean>
                       <cmis:propertyString displayName="cmis:versionLabel" localName="cmis:versionLabel" propertyDefinitionId="cmis:versionLabel"
queryName="cmis:versionLabel">
          <cmis:value>latest</cmis:value>
        </cmis:propertyString>
                                    <cmis:propertyBoolean displayName="cmis:isVersionSeriesCheckedOut" localName="cmis:isVersionSeriesCheckedOut"
propertyDefinitionId="cmis:isVersionSeriesCheckedOut" queryName="cmis:isVersionSeriesCheckedOut">
          <cmis:value>false</cmis:value>
        </cmis:propertyBoolean>
                        <cmis:propertyString displayName="cmis:lastModifiedBy" localName="cmis:lastModifiedBy" propertyDefinitionId="cmis:lastModifiedBy"
queryName="cmis:lastModifiedBy"/>
        <cmis:propertyString displayName="cmis:createdBy" localName="cmis:createdBy" propertyDefinitionId="cmis:createdBy" queryName="cmis:createdBy"/>
                  <cmis:propertyString displayName="cmis:checkinComment" localName="cmis:checkinComment" propertyDefinitionId="cmis:checkinComment"
queryName="cmis:checkinComment"/>
        <cmis:propertyId displayName="cmis:objectId" localName="cmis:objectId" propertyDefinitionId="cmis:objectId" queryName="cmis:objectId">
          <cmis:value>f708e208c0a80003554babb97bd934ba</cmis:value>
        </cmis:propertyId>
                        <cmis:propertyBoolean displayName="cmis:isImmutable" localName="cmis:isImmutable" propertyDefinitionId="cmis:isImmutable"
queryName="cmis:isImmutable">
          <cmis:value>false</cmis:value>
        </cmis:propertyBoolean>
                     <cmis:propertyBoolean displayName="cmis:isMajorVersion" localName="cmis:isMajorVersion" propertyDefinitionId="cmis:isMajorVersion"
queryName="cmis:isMajorVersion">
          <cmis:value>false</cmis:value>
        </cmis:propertyBoolean>
        <cmis:propertyId displayName="cmis:baseTypeId" localName="cmis:baseTypeId" propertyDefinitionId="cmis:baseTypeId" queryName="cmis:baseTypeId">
          <cmis:value>cmis:document</cmis:value>
        </cmis:propertyId>
                                        <cmis:propertyString displayName="cmis:contentStreamFileName" localName="cmis:contentStreamFileName"
propertyDefinitionId="cmis:contentStreamFileName" queryName="cmis:contentStreamFileName">
          <cmis:value>test.txt</cmis:value>
        </cmis:propertyString>
         <cmis:propertyDateTime displayName="cmis:lastModificationDate" localName="cmis:lastModificationDate" propertyDefinitionId="cmis:lastModificationDate"
queryName="cmis:lastModificationDate">
          <cmis:value>2010-09-09T18:06:31.987Z</cmis:value>
        </cmis:propertyDateTime>
      </cmis:properties>
      <cmis:acl/>
```

```
      <cmis:exactACL>false</cmis:exactACL>
      <cmis:policyIds/>
      <cmis:rendition/>
    </cmisra:object>
  </entry>
```

Then, get the *test.txt* file content via CMIS by using the **file** service and **id** of the file *test.txt* from *childs.xml*:

```
curl -o test.txt -u root:gtn http://localhost:8080/rest/private/cmisatom/Managed%20Sites/file/f708e208c0a80003554babb97bd934ba
```

Get results in the *test.txt* file in the local folder. In this way, you will get file stored in the **Sites Explorer** to folder */acme/documents/test.txt* via **eXo CMIS** by symlink path */acme/categories/acme/News/test.txt*. As file's actual content referenced by id in CMIS, the path has no matter for content read or change.

2.4.1.4. Modify WCM via CMIS

Note

Upload the modified local file using the command with id of the file stored in WCM (it is jcr:uuid of the file in JCR Workspace). Note that you should run this command from the folder where the local file is stored.

To update any file via CMIS, use the **file** service and the PUT request. Use the same file as in the previous use case (*/acme/documents/test.txt*, id:f708e208c0a80003554babb97bd934ba).

```
curl -T test.txt -X PUT -H "Content-Type:text/plain; charset=UTF-8" -u root:gtn http://localhost:8080/rest/private/cmisatom/Managed%20Sites/file/f708e208c0a80003554babb97bd934ba
```

Go to the **Sites Explorer** to see changes applied from the local file in */acme/documents/test.txt*.

2.4.1.5. CMIS search

CMIS provides a type-based query service for discovering objects that match specified criteria by defining a read-only projection of the CMIS data model into a Relational View.

CMIS query languages are based on a subset of the SQL-92 grammar. CMIS-specific language extensions to SQL-92 are called out explicitly. The basic structure of a CMIS query is a SQL statement that MUST include the following clauses:

- SELECT (virtual columns): This clause identifies the set of virtual columns that will be included in the query results for each row.
- FROM (Virtual Table Names): This clause identifies which Virtual Table(s) the query will run against.

Additionally, a CMIS query MAY include the following clauses:

- WHERE (conditions): This clause identifies the constraints that rows MUST satisfy to be considered a result for the query.
- ORDER BY (sort specification): This clause identifies the order in which the result rows MUST be sorted in the result row set.

Each CMIS ObjectType definition has the following query attributes:

Name	Description
query name (String)	Used for query operations on object types. In our SQL statement examples, all objecttypes is a queryName. For example, the given queryName matches the specific type of document. For example, in query like "SELECT * FROM cmis:document" ,"cmis:document" is queryName preconfigured in Document object type definition.
queryable (Boolean)	Indicate whether or not this object type is queryable. A non-queryable object type is not visible through the relational view that is used for query, and can not appear in the FROM clause of a query statement.
fulltextIndexed (Boolean)	Indicate whether objects of this type are full-text indexed for querying via the CONTAINS() query predicate.
includedInSupertypeQuery (Boolean)	Indicate whether this type and its subtypes appear in a query of this type's ancestor types. For example, if Invoice is a sub-type of Document, and its value is TRUE for a query on Document type, the matched instances of Invoice will be returned. If this attribute is FALSE, no instances (including matched ones) of Invoice will be returned.

Property definition also contains queryName and queryable attributes with the same usage.

2.4.1.5.1. Query examples

This section gives query examples for each specific case, including:

- Simple query
- Find document by several constraints
- Full-text search
- Extended full-text search
- Date property comparison
- Boolean property comparison
- IN Constraint
- Select all documents where longprop property is not in set
- Select all documents where longprop property is not in set
- IN_FOLDER constraint
- Select all documents that are in a specified folder
- Select all documents where query supertype is cmis:article
- IN_TREE constraint
- LIKE Comparison
- LIKE constraint with escape symbols
- NOT constraint
- Property existence
- ORDER BY
- ORDER BY ASC
- ORDER BY DESC
- ORDER BY SCORE (as columns)

- Not equal comparison (decimal)
- Not equal comparison (string)
- More than comparison (>)

2.4.1.5.1.1. Simple query

Query: Select all cmis:document.

```
SELECT * FROM cmis:document
```

Query result:

- All documents from the Apollo program.

2.4.1.5.1.2. Find document by several constraints

Query: Select all documents where apollo:propertyBooster is 'Saturn V' and apollo:propertyCommander is Frank F. Borman, II or James A. Lovell, Jr.

Initial data:

- document1: apollo:propertyBooster - Saturn 1B, apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyBooster - Saturn V, apollo:propertyCommander - Frank F. Borman, II
- document3: apollo:propertyBooster - Saturn V, apollo:propertyCommander - James A. Lovell, Jr.

```
SELECT * FROM cmis:document WHERE apollo:propertyBooster = 'Saturn V' AND (apollo:propertyCommander = 'Frank F. Borman, II' OR  apollo:propertyCommander = 'James A. Lovell, Jr.')
```

Query result:

- document2 and document3.

2.4.1.5.1.3. Full-text search

Query: Select all documents that contains the "here" word.

Initial data:

- document1: content - "There must be test word"
- document2: content - "Test word is not here"

```
SELECT * FROM cmis:document WHERE CONTAINS('here')
```

Query result:

- document2.

2.4.1.5.1.4. Extended full-text search

Query: Select all documents that contains "There must" phrase and do not contain the "check-word" term.

Initial data:

- document1: content - "There must be test word."
- document2: content - "Test word is not here. Another check-word."
- document3: content - "There must be check-word."

```
SELECT * FROM cmis:document WHERE CONTAINS("There must" - "check-word")
```

Query result:

- document1.

2.4.1.5.1.5. Date property comparison

Query: Select all documents where cmis:lastModificationDate is more than 2007-01-01.

Initial data:

- document1: cmis:lastModificationDate - 2006-08-08
- document2: cmis:lastModificationDate - 2009-08-08

```
SELECT * FROM cmis:document WHERE (cmis:lastModificationDate >= TIMESTAMP '2007-01-01T00:00:00.000Z')"
```

Query result:

- document2.

2.4.1.5.1.6. Boolean property comparison

Query: Select all documents where property apollo:someProperty equals to false.

Initial data:

- document1: apollo:someProperty - true
- document2: apollo:someProperty - false

```
SELECT * FROM cmis:document WHERE (apollo:someProperty = FALSE)
```

Query result:

- document2.

2.4.1.5.1.7. IN Constraint

Query: Select all documents where apollo:propertyCommander is in set {'Virgil I. Grissom', 'Frank F. Borman, II', 'James A. Lovell, Jr.'}.

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. Borman, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. Cernan

```
SELECT * FROM cmis:document WHERE apollo:propertyCommander IN ('Virgil I. Grissom', 'Frank F. Borman, II', 'James A. Lovell, Jr.')
```

Query result:

- document2, document3.

2.4.1.5.1.8. Select all documents where longprop property is not in set

Query: Select all documents where apollo:propertyCommander property is not in set {'Walter M. Schirra', 'James A. Lovell, Jr.'}.

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. Borman, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. Cerna

```
SELECT * FROM cmis:document WHERE apollo:PropertyCommander NOT IN ('Walter M. Schirra', 'James A. Lovell, Jr.')
```

Query result:

- document2, document4.

2.4.1.5.1.9. Select all documents where longprop property is not in set

Query: Select all documents where *apollo:propertyCommander_ property is not in set {'James A. Lovell, Jr.'}.*

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. Borman, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. Cerna

```
SELECT * FROM cmis:document WHERE  NOT (apollo:propertyCommander NOT IN ('James A. Lovell, Jr.'))
```

Query result:

- document3.

2.4.1.5.1.10. IN_FOLDER constraint

Query: Select all folders that are in folder1.

Initial data:

- folder1: id - 123456789
  - document1: Title - node1
- folder3:
  - folder4:

- folder2:
  - document2: Title - node2

```
SELECT * FROM cmis:folder WHERE IN_FOLDER('123456789')
```

Query result:

- folder3.

2.4.1.5.1.11. Select all documents that are in a specified folder

Query: Select all documents that are in folder1.

Initial data:

- folder1: id - 123456789
  - document1: Title - node1
- folder2:
  - document2: Title - node2

```
SELECT * FROM cmis:document WHERE IN_FOLDER('123456789')
```

Query result:

- document1.

2.4.1.5.1.12. Select all documents where query supertype is cmis:article

Initial data:

- testRoot: id - 123456789
- document1: Title - node1 typeID - cmis:article-sports
- document2: Title - node2 typeID - cmis:article-animals

```
SELECT * FROM cmis:article WHERE IN_FOLDER('123456789')
```

Query result:

- document1, document2.

2.4.1.5.1.13. IN_TREE constraint

Query: Select all documents that are in the tree of folder1.

Initial data:

- folder1: id - 123456789
  - document1
- folder2:
  - document2

```
SELECT * FROM cmis:document WHERE IN_TREE('123456789')
```

Query result:

- document1, document2.

2.4.1.5.1.14. LIKE Comparison

Query: Select all documents where apollo:propertyCommander begins with "James".

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. James, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. James

```
SELECT * FROM cmis:document AS doc WHERE apollo:PropertyCommander LIKE 'James%'
```

Query result:

- document3.

2.4.1.5.1.15. LIKE constraint with escape symbols

Query: Select all documents where apollo:someProperty like 'ad%min%'.

Initial data:

- document1: Title - node1, apollo:someProperty - ad%min master
- document2: Title - node2, apollo:someProperty - admin operator
- document3: Title - node2, apollo:someProperty - radmin

```
SELECT * FROM cmis:document AS doc WHERE apollo:someProperty  LIKE 'ad%min%'
```

Query result:

- document1.

2.4.1.5.1.16. NOT constraint

Query: Select all documents that do not contain the "world" word.

Initial data:

- document1: Title - node1, content - hello world
- document2: Title - node2, content - hello

```
SELECT * FROM cmis:document WHERE NOT CONTAINS('world')
```

Query result:

- document2.

2.4.1.5.1.17. Property existence

Query: Select all documents that has apollo:propertyCommander property is NOT NULL.

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander -
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander -

```
SELECT * FROM cmis:document WHERE apollo:propertyCommander is NOT NULL
```

Query result:

- document1, document3.

2.4.1.5.1.18. ORDER BY

Query: Select all documents in default order (by document name).

Initial data:

- document1: Title - Apollo 7
- document2: Title - Apollo 8
- document3: Title - Apollo 13
- document4: Title - Apollo 17

```
SELECT cmis:lastModifiedBy, cmis:objectId, cmis:lastModificationDate FROM cmis:document
```

Query result:

- document3, document4, document1, document2.

2.4.1.5.1.19. ORDER BY ASC

Query: Order by apollo:propertyCommander property value (in ascending order).

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. Borman, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. Cerna

```
SELECT cmis:lastModifiedBy, cmis:objectId, cmis:lastModificationDate FROM cmis:document ORDER BY apollo:propertyCommander
```

Query result:

- document4, document2, document3, document1.

2.4.1.5.1.20. ORDER BY DESC

Query: Order by apollo:propertyCommander property value (in descending order).

Initial data:

- document1: apollo:propertyCommander - Walter M. Schirra
- document2: apollo:propertyCommander - Frank F. James, II
- document3: apollo:propertyCommander - James A. Lovell, Jr.
- document4: apollo:propertyCommander - Eugene A. James

```
SELECT cmis:lastModifiedBy, cmis:objectId, cmis:lastModificationDate FROM cmis:document ORDER BY cmis:propertyCommander DESC
```

Query result:

- document1, document3, document2, document4.

2.4.1.5.1.21. ORDER BY SCORE (as columns)

Query: Select all documents which contains word "moon" ordered by score.

Initial data:

- document1: content - "Earth-orbital mission, the first manned launch"
- document2: content - "from another celestial body - Earth's Moon"
- document3: content - "NASA intended to land on the Moon, but a mid-mission technical"
- document4: content - "It was the first night launch of a U.S. human"

```
SELECT cmis:lastModifiedBy, cmis:objectId, cmis:lastModificationDate FROM cmis:document WHERE CONTAINS('moon') ORDER BY SCORE()
```

Query result:

- document2, document3.

2.4.1.5.1.22. Not equal comparison (decimal)

Query: Select all documents which have apollo:propertyBooster peroperty that does not equal to 3.

Initial data:

- document1: Title - node1, apollo:propertyBooster - 3
- document2: Title - node2, apollo:propertyBooster - 15

```
SELECT * FROM cmis:document WHERE apollo:propertyBooster <> 3
```

Query result:

- document2.

2.4.1.5.1.23. Not equal comparison (string)

Query: Select all documents property apollo:someProperty that does not equals to "test word second".

Initial data:

- document1: apollo:someProperty - "test word first"
- document2: apollo:someProperty - "test word second"

```
SELECT * FROM cmis:document WHERE apollo:someProperty <> 'test word second'
```

Query result:

- document1.

2.4.1.5.1.24. More than comparison (>)

Query: Select all documents which have apollo:propertyBooster property which is more than 5.

Initial data:

- document1: apollo:propertyBooster - 3
- document2: apollo:propertyBooster - 15

```
SELECT * FROM cmis:document WHERE apollo:propertyBooster > 5
```

Query result:

- document2.

The CMIS specification prescribes:

- Domain model
- Services

2.4.2. CMIS Domain Model

The CMIS Domain Model defines a repository as a container and an entry point to the objects that is quite simple and non-restrictive. The followings are some of the common entities of the domain model.

- Repository is a container of objects with a set of "capabilities" which may be different depending on the implementation.
- Object is the entity managed by a CMIS repository.
- Object Type is a classification related to an object. It specifies a fixed and non-hierarchical set of properties ("schema") that all objects of that type have.
- Document Object is an elementary information entity.
- Folder Object is a collection of fileable objects.
- Relationship Object is used to describe a dependent object semantically.
- Policy Object represents an administrative policy applied to an object.
- Access Object defines permissions.
- Versioning is to support versioning for Document objects.
- Query is type-based in a simplified SQL SELECT statement.

- Change Log is a mechanism which enables applications to discover changes to the objects stored.

Note

CMIS specifies a query language based on the SQL-92 standard, plus the extensions, in conjunction with the model mapping defined by the CMIS's relational view.

All objects are classified by an Object Type which declares that all objects of the given type have some sets of properties in common. Each property consists of a set of attributes, such as the TypeID, the property ID, its display name, its query name, and more. There are only four base types, including Document, Folder, Relationship, and Policy. Also, you can extend those basic types by modifying a set of their properties.

Document Object and Folder Object are self-explanatory. Document Object has properties to hold document metadata, such as the document author, modification date and custom properties. It can also contain a content stream whether it is required, and renditions, such as a thumbnail view of document. Folder is used to contain objects. Apart from the default hierarchical structure, CMIS can optionally store objects in multiple folders or in no folders at all (so-called multifiling and unfiling capabilities). Relationship Object denotes the connection between two objects (target and source). An object can have multiple relationships with other objects. Policy Object is a way to define administrative policies in managing objects. For example, you can use a CMIS policy to define which documents are subject to retention policies.

2.4.3. CMIS Services

CMIS provides a set of services to access and manage the content or repository. These services include:

Name	Description
Repository Services	Discover information about the repository and the object types defined for the repository.
Navigation Services	Traverse the folder hierarchy in a CMIS repository, and to locate documents which are checked out.
Object Services	Execute ID-based CRUD functions (Create, Retrieve, Update, Delete) on objects in a repository.
Multi-filing Services (optional)	Put an object in more than one folder (multi-filing), or outside the folder hierarchy (unfiling).
Discovery Services	Search for queryable objects in a repository.
Versioning Services	Check out, navigate to documents, or update a Document Version Series (checkOut, cancelCheckOut, getPropertiesOfLatestVersion, getAllVersions, deleteAllVersions).
Relationship Services (optional)	Retrieve an object for its relationships.
Policy Services (optional)	Apply, remove, or query for policies.
ACL Services	Return and manage the Access Control List (ACL) of an object. ACL Services are not supported by all repositories.

Some repositories might not implement certain optional capabilities, but they are still considered as CMIS-compliant. Each service has binding which defines the way messages will be serialized and wired. Binding is based on HTTP and uses the Atom Publishing Protocol.

2.5. Service JARs

Note

JCRs are listed below for the informational purpose only. For example, if the customer application needs to use the same third-parties used by eXo CMIS but with another version. All files are delivered in eXo Platform 3.5 distribution.

eXo CMIS consists of JAR files below:

xCMIS project files (for xCMIS 1.2.x)

- xcmis-renditions-X.X.X.jar
- xcmis-restatom-X.X.X.jar
- xcmis-search-model-X.X.X.jar
- xcmis-search-parser-cmis-X.X.X.jar
- xcmis-search-service-X.X.X.jar
- xcmis-spi-X.X.X.jar

eXo WCM Storage Provider for xCMIS SPI (uses the same version as the WCM/ECMS system)

- exo-ecms-ext-xcmis-sp-Y.Y.Y.jar

Third-party dependencies (for xCMIS 1.2.x)

- *abdera-client-0.4.0-incubating.jar*
- *abdera-core-0.4.0-incubating.jar*
- *abdera-i18n-0.4.0-incubating.jar*
- *abdera-parser-0.4.0-incubating.jar*
- *abdera-server-0.4.0-incubating.jar*
- *antlr-runtime-3.1.3.jar*
- *axiom-api-1.2.5.jar*
- *axiom-impl-1.2.5.jar*
- *jaxen-1.1.1.jar*
- *lucene-regex-2.4.1.jar*

See also

- Overview
- CMIS specification
- xCMIS project
- CMIS features

Configuration

This chapter describes configurations used in Content. It consists of the following main sections:

- **Component**

  Description of the services which provide low-level functionality for UI components.
- **External Component Plugins**

  Description of the main external component plugins used in the Content function. Each part supplies an example configuration with the explanation about init-params so you can know how to use these plugins.
- **CMIS configuration**

  Description of CMIS configuration and other additional information.

3.1. Components

- **ActionServiceContainer**

  Manage actions (adding, removing, or executing actions, and more) in the system.
- **ApplicationTemplateManagerService**

  Manage dynamic Groovy templates for WCM-based products.
- **FragmentCacheService**

  Cache the response fragments which are sent to end-users.
- **JodConverterService**

  Convert documents into different office formats.
- **LiveLinkManagerService**

  Check broken links, updates links when the links are edited and extracts links to return a list of all links.
- **LockService**

  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.
- **NewsletterInitializationService**

  Initiate some data for the newsletter portlet.
- **NewsletterManagerService**

  Send newsletters to subscribers.
- **SiteSearchService**

  Allow finding all information matching with your given keyword.
- **SEOService**

  Help users manage SEO data of a page or a content, so their websites can achieve higher rankings on search engines.
- **QueryService**

  Manage many queries, including adding, removing or executing a query.
- **TaxonomyService**

Sort documents to ease searches when browsing documents online.

- **ThumbnailService**

  Resize all images into different sizes. Besides the default sizes, it also allows users to customize the images into the desired sizes.

- **TimelineService**

  Allow documents to be displayed by days, months and years.

- **WatchDocumentService**

  Allow watching/unwatching a document. If you are watching the document, you will receive a notification mail when there are any changes on the document.

- **WCMService**

  Allow 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.

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

See also

- External Component Plugins
- CMIS configuration

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. This component is enabled by default. 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>port</name>
      <value>${jodconverter.portNumbers}</value>
    </value-param>
    <value-param>
      <name>officeHome</name>
      <value>${jodconverter.officeHome}</value>
    </value-param>
    <value-param>
      <name>taskQueueTimeout</name>
      <value>${jodconverter.taskQueueTimeout}</value>
    </value-param>
    <value-param>
      <name>taskExecutionTimeout</name>
      <value>${jodconverter.taskExecutionTimeout}</value>
    </value-param>
    <value-param>
      <name>maxTasksPerProcess</name>
      <value>${jodconverter.maxTasksPerProcess}</value>
    </value-param>
    <value-param>
      <name>retryTimeout</name>
      <value>${jodconverter.retryTimeout}</value>
    </value-param>
  </init-params>
 </component>
```

Details:

- **Value-param**:

Name	Type	Value	Description
port	`Integer`	${jodconverter.portNumbers}	The number of ports to connect with the office server.
officeHome	`String`	${jodconverter.officeHome}	The absolute path to the office home on the current local computer.
taskQueueTimeout	`Long`	${jodconverter.taskExecutionTimeout}	The maximum living time of a task in the conversation queue.
taskExecutionTimeout	`Long`	${jodconverter.taskExecutionTimeout}	The maximum time to process a task.
maxTasksPerProcess	`Integer`	${jodconverter.maxTasksPerProcess}	The maximum number of tasks are processed.
retryTimeout	`Long`	${jodconverter.retryTimeout}	The interval time to try to restart the office services in case they unexpectedly stop.

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 example about the server configuration: http// :localhost:8080.
	`hostName`	`String`	`localhost`	
	`port`	`The port number`	`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*.

Note

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>
          <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 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*.

Note

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>
        <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`	

Field	Type	Value	Description
		`NOFOLLOW`	Allow search engines to follow links from a particular page to find other pages or not.
frequency	`ArrayList`	`Always`	Define how often a particular page is updated.
		`Hourly`	
		`Daily`	
		`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	

Name	Type	Value	Description
			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.TaxonomyTreeDefaultUserPermission$Permission">
                <field name="identity">
                  <string>*:/platform/administrators</string>
                </field>
                <field name="read">
                  <string>true</string>
                </field>
                <field name="addNode">
                  <string>true</string>
                </field>
                <field name="setProperty">
                  <string>true</string>
                </field>
                <field name="remove">
                  <string>true</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyTreeDefaultUserPermission$Permission">
                <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`	*{java.util.ArrayList}*	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/bmp`	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,

Properties-param	Property name	Type	Value	Description
				the cache is cleared every 30 seconds.

3.2. External Component Plugins

- **AuthoringPublicationPlugin**

  Manage the publication lifecycle of web contents and DMS document on a portal page with more states and versions.
- **BPActionPlugin**

  Define the business process action in Workflow.
- **ContentTypeFilterPlugin**

  Filter the WCM node types.
- **ContextPlugin**

  Store the context configuration of a publication lifecycle.
- **ExcludeIncludeDataTypePlugin**

  Filter the search results before these results are presented on the search page.
- **FriendlyPlugin**

  Refine URLs in Content.
- **ImageThumnailPlugin**

  Configure the file types and get thumbnail for images.
- **IgnorePortalPlugin**

  Avoid deploying data to the existing portals during a new portal creation.
- **InitialWebcontentPlugin**

  Deploy initial web-contents as the site artifact into the Site Artifact folder of a newly created portal.
- **LinkDeploymentPlugin**

  Create predefined Symlinks into the system.
- **LockGroupsOrUsersPlugin**

  Configure predefined groups or users for lock administration.
- **ManageDrivePlugin**

  Create a predefined drive into a repository.
- **ManageViewPlugin**

  Create a predefined View into a repository.
- **PDFThumnailPlugin**

  Set the supported file types of PDF thumbnail.
- **PortetTemplatePlugin**

  Import the view templates into Content List Viewer.
- **predifinedProcessesPlugin**

  Import the predefined processes into the system.
- **QueryPlugin**

  Store predefined queries into the repositories of the system.

- **RemoveTaxonomyPlugin**

  Invalidate taxonomy trees in categories folder of a portal when the portal is removed.
- **ScriptActionPlugin**

  Import the predefined script actions into the system.
- **ScriptPlugin**

  Add groovy scripts into the system.
- **StageAndVersionPublicationPlugin**

  Control the state life cycle of a content.
- **StatesLifecyclePlugin**

  Control the state life cycle of a content.
- **TagPermissionPlugin**

  Configure the predefined permission for tags to inject in JCR.
- **TagStylePlugin**

  Configure the predefined styles for tags to inject in JCR.
- **TaxonomyPlugin**

  Configure the predefined taxonomies to inject into JCR.
- **TemplatePlugin**

  Create templates into the system. A template is a presentation to display the saved information.
- **XMLdeploymentPlugin**

  Deploy some "default" contents, such as Banner, Footer, Navigation, and Breadcrumb of a website.
- **BaseActionPlugin/CreatePortalPlugin/PublicationPlugin/RemovePortalPlugin**

  Create a link from the plugin to its children.

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

See also

- Components
- CMIS configuration

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. 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:validator=*:/platform/administrators
        </string>
      </field>
      <field name="mixins">
        <collection type="java.util.ArrayList">
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
              <field name="name">
                <string>exo:publishLocation</string>
              </field>
              <field name="properties">
                <string>
                  exo:publishWorkspace=collaboration;exo:publishPath=/Documents/Live;exo:validator=*:/platform/administrators
                </string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
              <field name="name">
                <string>exo:pendingLocation</string>
              </field>
              <field name="properties">
                <string>
                  exo:pendingWorkspace=collaboration;exo:pendingPath=/Documents/Pending
                </string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
              <field name="name">
                <string>exo:backupLocation</string>
              </field>
              <field name="properties">
                <string>exo:backupWorkspace=backup;exo:backupPath=/Expired
                  Documents
                </string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
              <field name="name">
                <string>exo:trashLocation</string>
              </field>
              <field name="properties">
                <string>
                  exo:trashWorkspace=collaboration;exo:trashPath=/Documents/Trash
                </string>
              </field>
            </object>
          </value>
          <value>
            <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
              <field name="name">
                <string>mix:affectedNodeTypes</string>
              </field>
              <field name="properties">
                <string>
                  exo:affectedNodeTypeNames=exo:article,exo:podcast,exo:sample,kfx:document,nt:file,rma:filePlan
                </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`	*{java.util.ArrayList}*	The action name.

3.2.3. 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`	`exo:cssFolder`	The folder type.
contentTypes	`ArrayList`	*{java.util.ArrayList}*	The content type.

3.2.4. 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.ContextConfig$Context">
                  <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.

3.2.5. 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/x-javascript,text/ecmascript`	The list of mimetypes which will be excluded from the search results.

3.2.6. FriendlyPlugin

This plugin is used to refine URLs in Content.

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.

3.2.7. 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.

3.2.8. IgnorePortalPlugin

When a new portal is created, the configuration of *IgnorePortalPlugin* is used to avoid deploying data to the existing ones which are listed in the init-parameters.

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/deployment/template-deployment-configuration.xml*.

Sample configuration:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.wcm.portal.artifacts.CreatePortalArtifactsService</target-component>
  <component-plugin>
   <name>Add ignored portals</name>
   <set-method>addIgnorePortalPlugin</set-method>
   <type>org.exoplatform.services.wcm.portal.artifacts.IgnorePortalPlugin</type>
   <description>ignored portals. the service will not deploy data to the ignored portals</description>
   <init-params>
    <values-param>
     <name>ignored.portals</name>
     <description>ignored portal list</description>
     <value>classic</value>
     <value>acme</value>
     <value>WAIPortal</value>
    </values-param>
   </init-params>
  </component-plugin>
</external-component-plugins>
```

In which:

- **Name**: `Add ignored portals`
- **Set-method**: `addIgnorePortalPlugin`
- **Type**: `org.exoplatform.services.wcm.portal.artifacts.IgnorePortalPlugin`

Init-params

Name	Type	Value	Description
ignored.portals	`string`	`classic, acme, WAIPortal`	The list of ignored existing portals.

3.2.9. 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 when
          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>
          <object-param>
            ...
          </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.

Name	Type	Value	Description
workspace	`string`	`collaboration`	The workspace into which the initial web contents will be deployed.
nodePath	`string`	`/sites content/live/{portalName}/web contents/site artifacts`	The target node where the initial web-contents will be deployed into.
sourcePath	`string`	`war:/conf/sample-portal/wcm/artifacts/site-resources/acme-templates/Logo.xml`	The path to the source that this plugin will get data.

3.2.10. 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.

3.2.11. 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`	*{java.util.ArrayList}*	The list of the groups or user to be locked.

3.2.12. 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 userId at runtime in the path.

Field	Type	Value	Description
permissions	`String`	*:/platform/administrators	Visibility of the drive based on eXo rights. For example: `*:/platform/users`
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: `simple-view,admin-view`
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: `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.

3.2.13. 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; showJCRStructure</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/artifacts	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.
permissions	`string`	*:/platform/administrators	Visibility of the view based on eXo rights.
template	`string`	`/exo:ecm/views/templates/ecm-explorer/SystemView`	Specify path to the template location.
tabList	`ArrayList`	*{java.util.ArrayList}*	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`	`/ecm-explorer/SystemView.gtmpl`	Specify a template location to view.

3.2.14. 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.

3.2.15. 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>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.

3.2.16. 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>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-${project.version}.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-bonita-payraise-${project.version}.jar</string>
              </value>
              <value>
                <string>/exo-ecms-ext-workflow-bp-bonita-holiday-${project.version}.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.

3.2.17. 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>CreatedDocumentDayBefore</string>
               </field>
               <field name="language">
                  <string>xpath</string>
               </field>
               <field name="statement">
                  <string>//element(*,exo:article)[@exo:dateCreated &lt; xs:dateTime('${Date}$')] 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>
      <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
`autoCreateInNewReposit`	`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.

3.2.18. 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`

3.2.19. 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$Action">
        <field name="type">
          <string>exo:sendMailAction</string>
        </field>
        <field name="name">
          <string>sendMail</string>
        </field>
        <field name="description">
          <string>send a notification mail</string>
        </field>
        <field name="srcWorkspace">
          <string>collaboration</string>
        </field>
        <field name="srcPath">
          <string>/Documents/Validation Requests</string>
        </field>
        <field name="isDeep">
          <boolean>true</boolean>
        </field>
        <field name="lifecyclePhase">
          <collection type="java.util.ArrayList">
            <value>
              <string>read</string>
            </value>
          </collection>
        </field>
        <field name="roles">
          <string>*:/platform/administrators</string>
        </field>
        <field name="variables">
          <string>exo:to=benjamin.mestrallet@exoplatform.com</string>
        </field>
        <field name="mixins">
          <collection type="java.util.ArrayList">
            <value>
              <object type="org.exoplatform.services.cms.actions.impl.ActionConfig$Mixin">
                <field name="name">
                  <string>mix:affectedNodeTypes</string>
                </field>
                <field name="properties">
                  <string>
                    exo:affectedNodeTypeNames=exo:article,exo:podcast,exo:sample,kfx:document,nt:file,rma:filePlan
                  </string>
                </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`	*{java.util.ArrayList}*	The list of the actions.

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

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

3.2.20. 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 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>
                <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/artifacts`	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.

3.2.21. 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 state
      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*

3.2.22. 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.LifecyclesConfig$Lifecycle">
                        <field name="name">
                           <string>lifecycle1</string>
                        </field>
                        <field name="publicationPlugin">
                           <string>Authoring publication</string>
                        </field>
                        <field name="states">
                           <collection type="java.util.ArrayList">
                              <value>
                                 <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig$State">
                                    <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.lifecycle.impl.LifecyclesConfig$State">
                                <field name="state">
                                    <string>pending</string>
                                </field>
                                <field name="membership">
                                    <string>author:/platform/web-contributors</string>
                                </field>
                            </object>
                        </value>
                        <value>
                            <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig$State">
                                <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.lifecycle.impl.LifecyclesConfig$State">
                                <field name="state">
                                    <string>staged</string>
                                </field>
                                <field name="membership">
                                    <string>publisher:/platform/web-contributors</string>
                                </field>
                            </object>
                        </value>
                        <value>
                            <object type="org.exoplatform.services.wcm.extensions.publication.lifecycle.impl.LifecyclesConfig$State">
                                <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.

Field	Type	Value	Description
states	`ArrayList`	*{java.util.ArrayList}*	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.

3.2.23. 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.

3.2.24. 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$HtmlTagStyle">
                <field name="name">
                  <string>attractive</string>
                </field>
                <field name="tagRate">
                  <string>5..7</string>
                </field>
                <field name="htmlStyle">
                  <string>font-size: 15px; font-weight: bold; color: blue; font-family: Arial;
                    text-decoration:none;
                  </string>
                </field>
                <field name="description">
                  <string>attractive style for tag</string>
                </field>
              </object>
            </value>
            <value>
              <object type="org.exoplatform.services.cms.folksonomy.impl.TagStyleConfig$HtmlTagStyle">
                <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: Arial;
                    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$HtmlTagStyle">
                <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:Arial;
                    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
autoCreatedInNewRepositor	`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.

3.2.25. 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$Taxonomy">
                      <field name="permissions">
                        <collection type="java.util.ArrayList">
                          <value>
                            <object type="org.exoplatform.services.cms.taxonomy.impl.TaxonomyConfig$Permission">
                              <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>
                  </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.

3.2.26. 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$NodeType">
                  <field name="nodetypeName">
                    <string>nt:file</string>
                  </field>
                  <field name="documentTemplate">
                    <boolean>true</boolean>
                  </field>
                  <field name="label">
                    <string>File</string>
                  </field>
                  <field name="referencedView">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                          <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$Template">
                          <field name="templateFile">
                            <string>/file/views/admin_view.gtmpl</string>
                          </field>
                          <field name="roles">
                            <string>*:/platform/administrators</string>
                          </field>
                        </object>
                      </value>
                    </collection>
                  </field>
                  <field name="referencedDialog">
                    <collection type="java.util.ArrayList">
                      <value>
                        <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                          <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.TemplateConfig$Template">
                            <field name="templateFile">
                              <string>/file/dialogs/admin_dialog.gtmpl</string>
                            </field>
                            <field name="roles">
                              <string>*:/platform/administrators</string>
                            </field>
                          </object>
                        </value>
                      </collection>
                    </field>
                    <field name="referencedSkin">
                      <collection type="java.util.ArrayList">
                        <value>
                          <object type="org.exoplatform.services.cms.templates.impl.TemplateConfig$Template">
                            <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.TemplateConfig$Template">
                            <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>
        </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/artifacts/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).

3.2.27. 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.zip
            </string>
          </field>
          <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.DeploymentDescriptor$Target (*)`	The target node where will contain the imported node.
sourcePath	`string`	`war:/conf/sample-portal/wcm/artifacts/site-resources/acme/Logo.xml<`	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.

Name	Type	Value	Description
			true: allow false:not allow
versionHistoryPath	`string`	`war:/conf/sample-portal/wcm/artifacts/site-resources/acme/Logo_versionHistory.zip<`	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 artifacts`	The path of the target node.

3.2.28. BaseActionPlugin/CreatePortalPlugin/PublicationPlugin/RemovePortalPlugin

All these plugins allows creating a link to its children.

BaseActionPlugin

This is an abstract class and the parent of ScriptActionPlugin and BPActionPlugin .

CreatePortalPlugin

This is an abstract class and the parent of *TaxonomyPlugin* .

PublicationPlugin

This is an abstract class and the parent of *StageAndVersionPublicationPlugin* and *AuthoringPublicationPlugin* .

RemovePortalPlugin

This is an abstract class and the parent of *RemoveTaxonomyPlugin* .

3.3. CMIS configuration

- **CMIS Configuration**

  Necessary information to make a special extension of *CmisRegistry*.
- **Required nodetypes and namespaces in JCR**

  Introduction to the mandatory configuration for JCR to work correctly.
- **Authenticator and organization service configuration**

  Overall introduction to configuration of authenticator and organization service.
- **CMIS search and index**

Full provision of CMIS and index, such as query capabilities, configuration, index atomicity and durability.

See also

- Components
- External Component Plugins

3.3.1. CMIS Configuration

To expose WCM drives to the CMIS repositories, you must make a special extension of *CmisRegistry*.

To make a typical component *org.exoplatform.ecms.xcmis.sp.jcr.exo.DriveCmisRegistry*, do as follows:

```
<component>
  <type>org.exoplatform.ecms.xcmis.sp.jcr.exo.DriveCmisRegistry</type>
  <init-params>
    <!-- Disabled by default. Uncomment if you need query support in CMIS. -->
      <!-- value-param>
        <name>indexDir</name>
        <value>${gatein.jcr.index.data.dir}/cmis-index${container.name.suffix}</value>
      </value-param-->
    <value-param>
      <name>exo.cmis.renditions.persistent</name>
      <value>true</value>
    </value-param>
    <values-param>
      <name>renditionProviders</name>
      <description>Redition providers classes.</description>
      <!-- <value>org.xcmis.renditions.impl.PDFDocumentRenditionProvider</value> -->
      <value>org.xcmis.renditions.impl.ImageRenditionProvider</value>
    </values-param>
  </init-params>
</component>
```

Where configuration parameters include:

- *indexDir*: the directory where the lucene index will be placed. It is disabled by default.
- *exo.cmis.renditions.persistent*: indicates if a rendition of the document (thumbnails) should be persisted in the JCR. The allowed value are *true* or *false*.
- *renditionProviders*: a set of FQN of classes which can be used as Rendition Providers. Classes which implement the *org.xcmis.spi.RenditionProvider* interface used to preview the CMIS objects (thumbnails).

Note

In most cases, it is not required to change anything in the xCIMIS configuration. In case of any change of the indexer storage location, do not comment the *indexDir* value parameter and point it to the actual location.

3.3.2. Required nodetypes and namespaces in JCR

The following configuration is mandatory for JCR to work correctly:

```
<external-component-plugins>
  <target-component>org.exoplatform.services.jcr.RepositoryService</target-component>
  <component-plugin>
    <name>add.namespaces</name>
    <set-method>addPlugin</set-method>
```

```
    <type>org.exoplatform.services.jcr.impl.AddNamespacesPlugin</type>
    <init-params>
      <properties-param>
        <name>namespaces</name>
        <property name="cmis" value="http://www.exoplatform.com/jcr/cmis/1.0"/>
        <property name="xcmis" value="http://www.exoplatform.com/jcr/xcmis/1.0"/>
      </properties-param>
    </init-params>
  </component-plugin>
  <component-plugin>
    <name>add.nodeType</name>
    <set-method>addPlugin</set-method>
    <type>org.exoplatform.services.jcr.impl.AddNodeTypePlugin</type>
    <init-params>
     <values-param>
        <name>autoCreatedInNewRepository</name>
        <description>Node types configuration file</description>
        <value>jar:/conf/cmis-nodetypes-config.xml</value>
      </values-param>
    </init-params>
  </component-plugin>
</external-component-plugins>
```

3.3.3. Authenticator and organization service configuration

An authenticator is responsible for creating Identity Security Service Authenticator.

The eXo CMIS service is based on:

- The authentication mechanism provided by the eXo organization service.
- The JAAS configuration of eXo Platform. For example:

```
gatein-domain {
  org.exoplatform.web.security.PortalLoginModule required;
  org.exoplatform.services.security.jaas.SharedStateLoginModule required;
  org.exoplatform.services.security.j2ee.TomcatLoginModule required;
};
```

3.3.4. CMIS search and index

The CMIS standard defines a query language based on a subset of the SQL-92 grammar (ISO/IEC 9075: 1992 -- Database Language SQL), with a few extensions to enhance its filtering capability for the CMIS data model, such as existential quantification for multi-valued property, full-text search, and folder membership.

Warning

CMIS search is disabled by default in eXo CMIS. Uncomment the *indexDir* parameter if you need the query support in CMIS. To discover the search capability, check the *capabilityQuery* property (see the table [105] below).

CMIS Relational View

The relational view of a CMIS repository consists of a collection of virtual tables that are defined on the top of the CMIS data model. A virtual table exists for every *queryable* object type (content type if you prefer) in the repository. Each row in these virtual tables corresponds to an instance of the corresponding object type (or one of its subtypes). A column exists for every property that the object type has.

Query Capabilities

Capability	Value
capabilityQuery	`bothcombined` (if *indexDir* is configured, otherwise `none`)
capabilityJoin	`none`
capabilityPWCSearchable	`false`
capabilityAllVersionsSearchable	`false`

Configuration

To be able to provide full-text search capabilities, xCMIS uses its own index. The following is the configuration parameter:

Parameter	Default	Description
indexDir	none	The location of the index directory. This parameter is mandatory for the default implementation.

This parameter is passed through the XML configuration.

For example, to set up the index directory:

```
<component>
  <type>org.exoplatform.ecms.xcmis.sp.DriveCmisRegistry</type>
  <init-params>
      <!-- Disabled by default. Uncomment if you need query support in CMIS. -->
      <!-- value-param>
        <name>indexDir</name>
        <value>${gatein.jcr.index.data.dir}/cmis-index${container.name.suffix}</value>
      </value-param-->
      ..............
  </init-params>
</component>
```

Index atomicity and durability

- **Write-ahead logging**

To be able to provide index consistency and recovery in case of unexpected crashes or damages, XCMIS uses write-ahead logging (WAL) technique. Write-ahead logging is a standard approach to transaction logging. Briefly, WAL's center concept is changes of data files (indexes) must be written only after those changes have been logged, that is, when the change log records have been flushed to permanent storage. If you follow this procedure, you do not need to flush data pages to disk on every transaction commit, because it is known in the event of a crash, and the index can be recovered by using the log: any changes that have not been applied to the data pages can be redone from the log records. (This is roll-forward recovery, also known as REDO.)

A major benefit of using WAL is a significantly reduced number of disk writes, because only the log file needs to be flushed to disk at the time of transaction commit, rather than every data file changed by the transaction.

- **Recover uncommitted transaction**

When you start Indexer, it will check uncommitted transaction logs. If at least one log exists, recovering process will be started. Indexer will read all logs and extract added, updated and removed UUIDs into a set. Then, indexer walks through this set and checks objects against UUID. If the object exists, the indexer will put it into the added document list. In other cases, UUID will be added to the removed documents list. After that, depending on the list of added and removed documents, changes will be applied to the index.

- **Initial index population**

When you run the indexer to check the number of documents in the index. If there are no documents in the index or the previous re-indexation was not successful, then re-indexation of all content will be started. The first step is cleaning old index

data. Uncommitted transaction logs and old persistent data are removed. These data are useless, because re-indexation of all content will be started. Then indexer walks throw all objects and makes lucene document for each one. Then batches with less than 100 elements will be saved to the index. After re-indexation, all logs (WAL) will be removed, all data mentioned on these change logs are already indexed.

Note

If the administrator gets an exception with the message "Can't remove reindex flag.", it means that the index restoring was finished but file-flag was not removed (see index directory, file named as "reindexProcessing"). You can manually remove this file-flag, and avoid a new reindex of repository on the JCR start.

Developer references

This chapter supplies you with the basic knowledge of templates, UI extension, APIs in Content. Throughout this chapter, you can build your own content. Also, with the step-by-step instructions, you can create UI extension, deploy a workflow in Content and do many different actions.

This chapter consists of the main following contents:

- **WCM Templates**

  Information about Content types and a list of Contents used in the Content function.
- **WCM Explorer**

  Introduction to the CSS files and CKEditor applied into the Content function.
- **Extensions**

  Knowledge of extensions in the Content function, such as REST services, UI extensions, authoring extensions and more.
- **CMIS Usage code examples**

  Examples of the CMIS usage code which may be useful for developers who need to access a repository.
- **Public REST APIs**

  Information (such as name, service URL, parameter, and description) of Public REST APIs used in the Content function.
- **Public Java APIs**

  Information about public Java APIs used in the Content function.
- **Deprecated portlets**

  A list of deprecated portlets in the Content function.
- **Miscellaneous and Tips**

  A list of reference links for xCMIS project and CMIS.
- **FAQs**

  A list of FAQs related to the product.

4.1. WCM Templates

- **Content types**

  Details of 2 template types (dialog and view) applied to a node type or a metadata mixin type.
- **List of Contents**

  Description about Content List and Category Navigation templates which are commonly used in Content.

See also

- WCM Explorer
- Extensions
- CMIS Usage code examples

- Public REST APIs
- Public Java APIs
- Deprecated portlets
- Miscellaneous and Tips
- FAQs

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. **Sites 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:title`	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: *name, email, number, empty, null, datetime, length* OR validator classes. To know how to

Parameter	Type	Required	Example	Description
				pass parameters to validators, refer here [111]
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.
options	`String` separated by the commas (,) character.		A list of parameters which are input while the content templates are initialized.	`"options=toolbar:CompleteWCM`

Pass parameters to validators

- "name" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=name", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "email" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=email", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "number" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=number", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "empty" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=empty", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "null" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=null", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "datetime" validator:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=datetime", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

- "length" validator:

For a maximum length of 50 characters:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=empty,length(50;int)", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

For a minimum length of 6 characters and maximum length of 50 characters:

```
String[] webContentFieldTitle = ["jcrPath=/node/exo:title", "validate=empty,length(6;50;int)", "editable=if-null"];
uicomponent.addTextField("title", webContentFieldTitle) ;
```

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:'1`
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 can be the percent of pixel.	`options=height:'200px'`

See also: Common parameters

- **Example**

```
<%
  String[] fieldSummary = ["jcrPath=/node/exo:summary", "options=toolbar:CompleteWCM,width:'100%',height:'200px'", "validate=empty"] ;
  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", "visible=true"] ;
  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.findComponentById(image) == null)) {
    def imgSrc = uicomponent.getImage(curNode, "exo:image");
    def actionLink = uicomponent.event("RemoveData", "/exo:image");
    %>
     <div>
      <img src="$imgSrc" width="100px" height="80px"/>
      <a href="$actionLink">
       <img src="/eXoResources/skin/DefaultSkin/background/Blank.gif" alt="" class="ActionIcon Remove16x16Icon"/>
      </a>
     </div>
    <%
   } else {
    String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
    uicomponent.addUploadField(image, fieldImage) ;
   }
  } else {
    String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
    uicomponent.addUploadField(image, fieldImage) ;
  }
 }  else if(uicomponent.dataRemoved()) {
  String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
  uicomponent.addUploadField(image, fieldImage) ;
 } else {
```

```
 String[] fieldImage = ["jcrPath=/node/exo:image/jcr:data"] ;
 uicomponent.addUploadField(image, fieldImage) ;
 }
%>
```

- To have multiple upload fields, you just add the **multiValues=true** parameter to **fieldProperty** in *dialog1.gtmpl*:

```
# Multi upload
fieldProperty = ["jcrPath=/node/exo:value", "multiValues=true"];
uicomponent.addUploadField("/node/exo_value", fieldProperty);
```

Note

In this case, you must be sure that the node type definition of the document you are currently editing should allow the document to have a child node named '*exo:value*' whose node type is '*nt:unstructured*'. All uploaded files of this upload component are stored in this '*exo:value*' child node.

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.exoplatfo`
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.UIOneNodePathSelector",
 "workspaceField=targetWorkspace", "selectorIcon=SelectPath24x24Icon"] ;
  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.
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("WebContent.dialog.label.MainContent")%></
div>
      </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.appRes("WebContent.dialog.label.Illustration")%></div>
   </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.appRes("WebContent.dialog.label.Advanced")%></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")%>">
```

4.1.1.1.15. How to prevent XSS attacks

In the content management sytem, its typical feature is enabling JavaScript in a content. This causes the XSS (Cross-site Scripting) attacks to the content displayed in the HTML format.

However, there is no solution to keep JavaScript and to prevent the XSS attacks at the same time, so Content allows you to decide whether JavaScript is allowed to run on a field of the content template or not by using the `option` parameter.

- To allow JavaScript to execute, add "`options = noSanitization`" to the dialog template file. Normally, this file is named `dialog1.gtmpl`.
- For example: The following code shows how to enable JavaScript in the **Main Content** field of the **Free Layout Wecontent** content:

```
String [] htmlArguments = ["jcrPath = / node / default.html / JCR: content / JCR: data", "options = toolbar: CompleteWCM, height: '410px ', noSanitization" htmlContent];
```

- By default, there is no "`options = noSanitization`" parameter in the dialog template file and this helps you prevent the XSS attacks. When end-users input JavaScript into a content, the JavaScript is automatically deleted when the content is saved.

4.1.1.2. View

The following is a 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("MMMMM dd yyyy") ;
%>
     <%=dateFormat.format(node.getProperty("exo:date").getDate().getTime())%>
<%
   }
%>
```

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

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

4.1.2. List of Contents

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.

Template	Description
ContentListViewerDefault.gtmpl	Its function is similar to *BigHotNewsTemplateCLV.gtmpl*. The illustration of each content is bigger.
DocumentsTemplate.gtmpl	Display contents under a content list with a NodeType icon or the illustration on the left of the corresponding content.
EventsTemplateCLV.gtmpl	Its function is similar to *BigHotNewsTemplateCLV.gtmpl*, 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 *BigHotNewsTemplateCLV.gtmpl*, but the illustration of each content is bigger than the image displayed with *ContentListViewerDefault.gtmpl* and the text font is different.
UIContentListPresentationDefault.gtmpl	Its function is similar to *BigHotNewsTemplateCLV.gtmpl*, 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.

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

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.

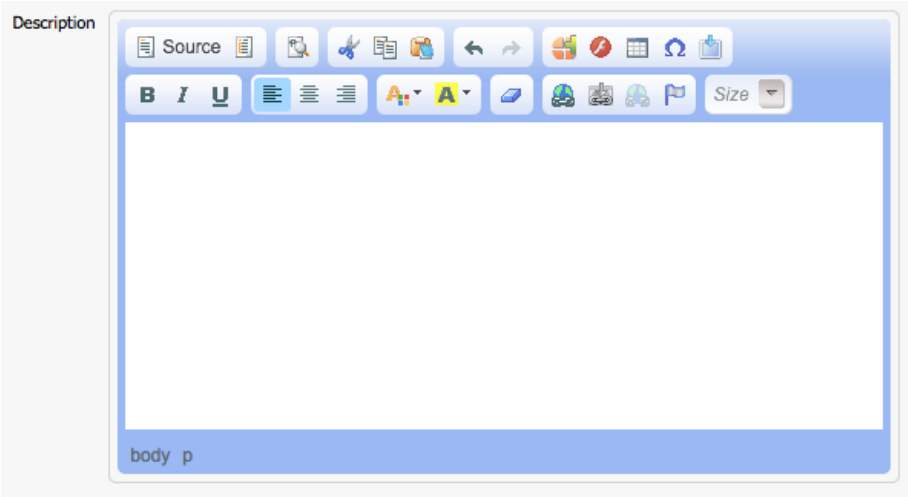
Note

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.

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>
```



See also

- WCM Templates
- Extensions
- CMIS Usage code examples
- Public REST APIs
- Public Java APIs
- Deprecated portlets
- Miscellaneous and Tips
- FAQs

4.3. Extensions

- **REST Services**

  Introduction to Rest Services, and details (including HTTP Methods, formats, data format, REST configuration) of Restful Web Service, and how to create a REST service.

- **How to make your own ECMS UI Extensions**

  Instructions on how to make your own UI extension by creating the new action and its corresponding listener, registering with *UIExtensionManager* and running your UI extension sample. This section also provides information on filtering action with exsiting filters or creating your new filter.

- **Authoring Extension**

  Information about the extended publication plugin and publication manager.

- **Auxiliary attributes for documents**

  Details of how to create the *DocumentContext* which stores some auxiliary attributes of the document and helps document listeners make decision based on these attributes.

See also

- WCM Templates
- WCM Explorer
- CMIS Usage code examples
- Public REST APIs
- Public Java APIs
- Deprecated portlets
- Miscellaneous and Tips
- FAQs

4.3.1. REST Services

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.

4.3.1.1. Restful Web Service

This section provides you the following topics:

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.

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.

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.

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.exoplatform.services.ecm.publication.REST.presentation.document.publication.PublicationGetDocumentRESTService</type>
  </component>
</configuration>
```

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 {
    ........
  }
}
```

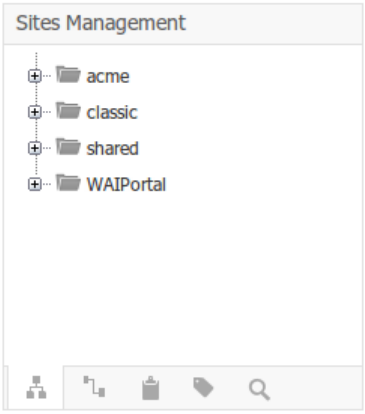
Parameters	Definition
@Path("/presentation/document/edit/")	Specify the URI path which a resource or class method will serve requests for.
@PathParam("repository")	Bind the value **repository** of a URI parameter or a path segment containing the template parameter to a resource method parameter, resource 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. How to make your own ECMS UI Extensions

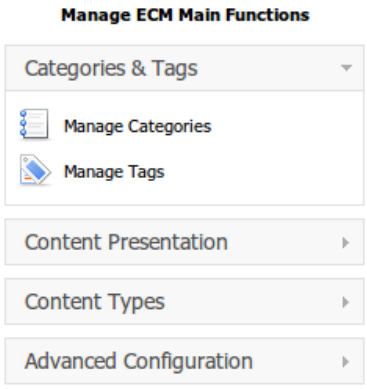
There are many places inside ECMS which are built basing on the UI Extensions framework as described in Extend eXo applications, so you can add your own actions packaged in external jars to them. They are:

- **Action bar**

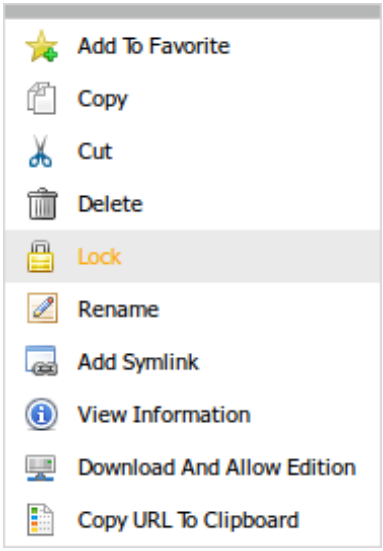
- **Side bar**



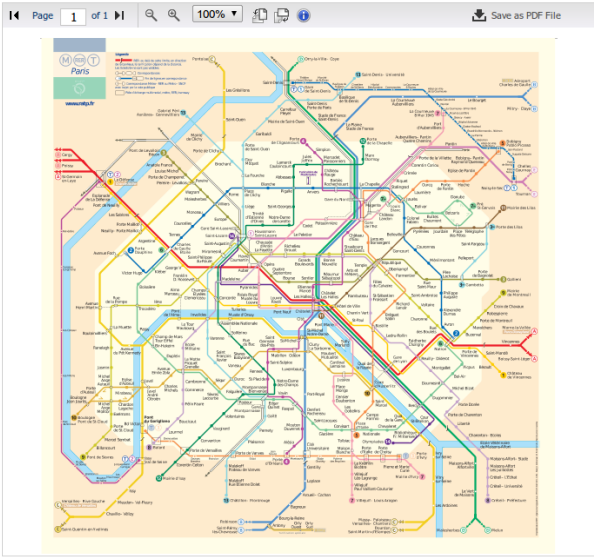
- **Admin control panel**



- **Context menu in the main working area**



- **File viewer**



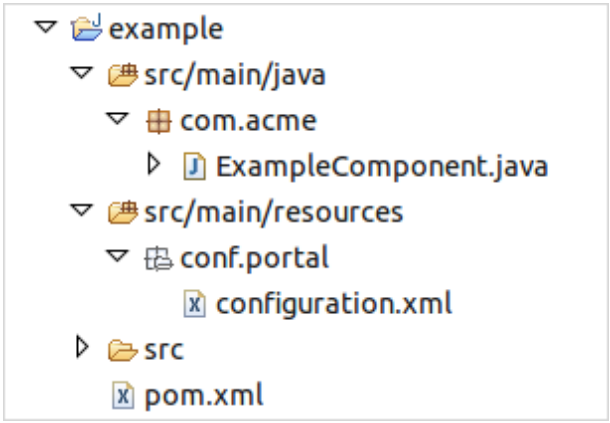
4.3.2.1. How to add an action extension

This section shows you how to add own actions to the ECMS. In the following example, you are going to add a new action on the ECMS action toolbar to view the node path.

Follow the below process to add a new action to the action toolbar:

Create a new project for action extension

Create a Maven project which has the following directory structure:



Navigating in the project's folder, you will see the following structure:

- *pom.xml*: The project's POM file.
- *src/main/java/.../ExampleActionComponent.java*: A simple action supporting user to view the wiki markup of a page.
- *src/main/resources/conf/portal/configuration.xml*: The configuration file to register your actions with the *org.exoplatform.webui.ext.UIExtensionManager* service.

Here is the content of the *pom.xml* file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.acme</groupId>
 <artifactId>example</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>example</name>
 <url>ECMS action example</url>

 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>
 <dependencies>
  <dependency>
   <groupId>org.exoplatform.portal</groupId>
   <artifactId>exo.portal.webui.core</artifactId>
   <version>3.2.5-PLF-SNAPSHOT</version>
   <scope>provided</scope>
  </dependency>
  <dependency>
   <groupId>org.exoplatform.commons</groupId>
   <artifactId>exo.platform.commons.webui.ext</artifactId>
   <version>1.1.9-SNAPSHOT</version>
   <scope>provided</scope>
  </dependency>
  <dependency>
   <groupId>org.exoplatform.ecms</groupId>
   <artifactId>exo-ecms-core-webui-explorer</artifactId>
   <version>2.3.8-SNAPSHOT</version>
  </dependency>
 </dependencies>
 <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
     <source>1.5</source>
     <target>1.5</target>
    </configuration>
   </plugin>
```

```
  </plugins>
 </build>
</project>
```

Create new action and its corresponding listener

Edit the *ExampleActionComponent* class as below:

```
package com.acme;

import javax.jcr.Node;

import org.exoplatform.ecm.webui.component.explorer.UIJCRExplorer;
import org.exoplatform.ecm.webui.component.explorer.control.listener.UIActionBarActionListener;
import org.exoplatform.web.application.ApplicationMessage;
import org.exoplatform.webui.config.annotation.ComponentConfig;
import org.exoplatform.webui.config.annotation.EventConfig;
import org.exoplatform.webui.core.UIComponent;
import org.exoplatform.webui.event.Event;

@ComponentConfig(
 events = { @EventConfig(listeners = ExampleComponent.ExampleActionListener.class)
})
public class ExampleComponent extends UIComponent {

  public static class ExampleActionListener extends UIActionBarActionListener<ExampleComponent> {
    @Override
    protected void processEvent(Event<ExampleComponent> event) throws Exception {
      UIJCRExplorer uiJCRExplorer = event.getSource().getAncestorOfType(UIJCRExplorer.class);
      Node node = uiJCRExplorer.getCurrentNode();
      event.getRequestContext()
          .getUIApplication()
          .addMessage(new ApplicationMessage("Node path:"  node.getPath(), null, ApplicationMessage.INFO));
    }
  }

}
```

Register new action with UIExtensionManager

Edit the *configuration.xml* file as below:

```
<configuration   xmlns="http://www.exoplatform.org/xml/ns/kernel_1_2.xsd"   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xsi:schemaLocation="http://
www.exoplatform.org/xml/ns/kernel_1_2.xsd http://www.exoplatform.org/xml/ns/kernel_1_2.xsd">

 <external-component-plugins>
  <target-component>org.exoplatform.webui.ext.UIExtensionManager</target-component>
  <component-plugin>
    <name>add.action</name>
    <set-method>registerUIExtensionPlugin</set-method>
    <type>org.exoplatform.webui.ext.UIExtensionPlugin</type>
    <init-params>
      <object-param>
        <name>Example</name>
        <object type="org.exoplatform.webui.ext.UIExtension">
          <field name="type">
            <string>org.exoplatform.ecm.dms.UIActionBar</string>
          </field>
          <field name="name">
            <string>Example</string>
          </field>
          <field name="component">
            <string>com.acme.ExampleComponent</string>
          </field>
        </object>
```

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

</configuration>
```

4.3.2.2. Deploy new action extension

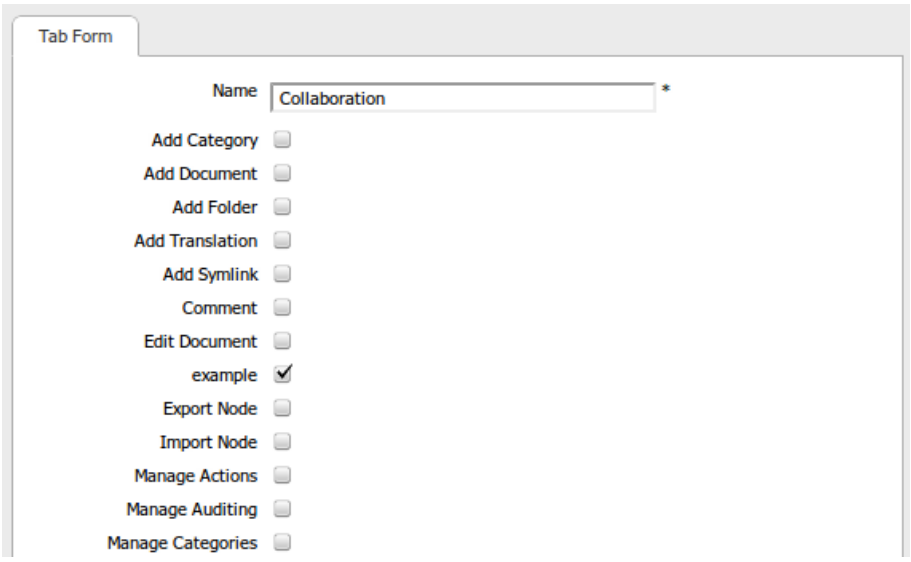
Follow these steps to deploy your new action extension:

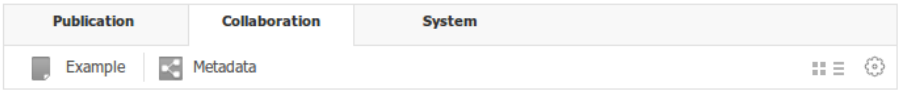
1. Build the project by using the *mvn clean install* command.

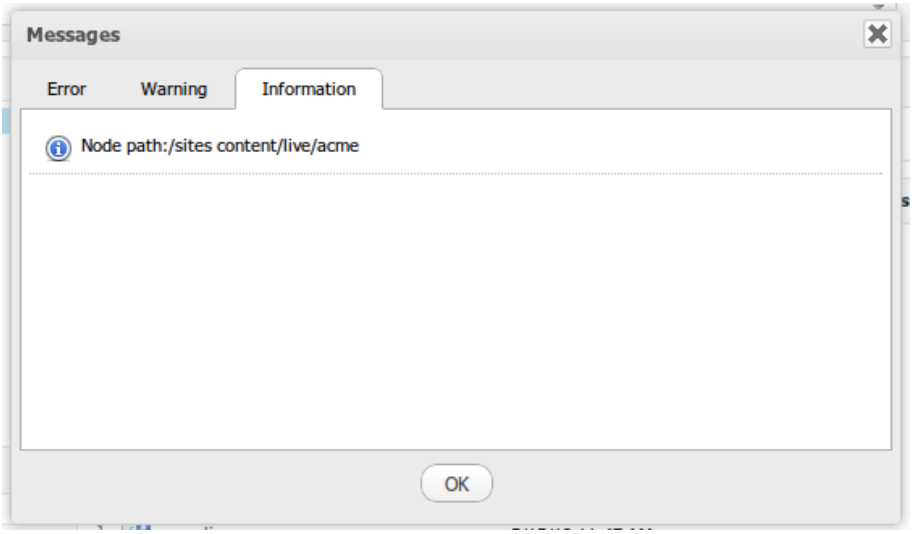
2. Copy the *target/example-1.0-SNAPSHOT.jar* file into the *TOMCAT-HOME/lib/* directory.

3. Run Tomcat and go to **Content explorer**.

4. Add the deployed action to the WCM View. You will get the results:







4.3.2.3. Define labels and logo

Define a label in ECM Administration

Edit the *$TOMCAT-HOME/webapps/ecmadmin/WEB-INF/classes/locale/portlet/administration/ECMAdminPortlet_en.xml* file (for English which is also the default language) and add the label as below:

```
...
 <UIViewFormTabPane>
   ...
  <label>
    <example>Example action</example>
   ...
  </label>
   ...
 </UIViewFormTabPane>
...
```

Define a label in the File Explorer

Edit the *$TOMCAT-HOME/webapps/ecmexplorer/WEB-INF/classes/locale/portlet/explorer/JCRExplorerPortlet_en.xml* file (for English which is also the default language) and add the label as below:

```
...
 <UIActionBar>
   ...
  <tooltip>
    <example>Example action</example>
   ...
  </tooltip>
   ...
 </UIActionBar>
...
```

Define Logos

Edit the *$TOMCAT-HOME/webapps/ecmexplorer/skin/icons/24x24/DefaultStylesheet.css* file (for the default Skin) and add the icon definition as below (in this case, the "ManageUnLock" icon is re-used but you could add your own picture into the *$TOMCAT-HOME/webapps/ecmexplorer/skin/icons/24x24/DefaultSkin* directory):

```
.ExampleIcon{
  width: 24px; height: 24px;
  background: url('DefaultSkin/ManageUnLock.gif') no-repeat left center; /* orientation=lt */
  background: url('DefaultSkin/ManageUnLock.gif') no-repeat right center; /* orientation=rt */
}
```

4.3.2.4. Filter your action

With the UI Extension framework, you can use internal and external (existing) filters. The internal filters are parts of the business logic of your component. For example, if your component is only dedicated to articles, you will add an internal filter to your component that will check the type of the current document. The external filters are mainly used to add new filters that are not related to the business logic, to your component. A good example is the *UserACLFilter* which allows you to filter by access permissions.

It is very simple to apply a filter in an UI component:

```
public class ExampleComponent extends UIComponent {

   private static final List<UIExtensionFilter> FILTERS = Arrays.asList(new UIExtensionFilter[] {new MyUIFilter()});

@UIExtensionFilters
public List<UIExtensionFilter> getFilters() {
return FILTERS;
}
...
```

4.3.2.4.1. Use existing filters

There are many useful built-in filters in ECMS:

- `org.exoplatform.webui.ext.filter.impl.UserACLFilter`: Filter all nodes that do not have any permission on the current context.
- `org.exoplatform.webui.ext.filter.impl.FileFilter`: Filter all nodes that do not exist in the given MIME type list.
- `org.exoplatform.ecm.webui.component.explorer.control.filter`: This package includes the following filters.

Filters	Description
`CanAddCategoryFilter`	Filter nodes to which it is impossible to add categories.
`CanCutNodeFilter`	Filter nodes which cannot be cut.
`CanAddNodeFilter`	Filter nodes to which it is impossible to add nodes.
`CanDeleteNodeFilter`	Filter nodes that cannot be deleted.
`CanRemoveNodeFilter`	Filter nodes that cannot be removed.
`CanEnableVersionFilter`	Filter nodes which do not allow versioning.
`CanSetPropertyFilter`	Filter nodes that cannot be modified.
`HasMetadataTemplatesFilter`	Filter nodes that do not have metadata templates.
`HasPublicationLifecycleFilter`	Filter all nodes that do not have the publication plugins.
`HasRemovePermissionFilter`	Filter nodes that do not have the **Remove** permission.
`IsFavouriteFilter`	Filter nodes that are not favorite.
`IsNotFavouriteFilter`	Filter nodes that are favorite.
`IsNotNtFileFilter`	Filter nodes that are of *nt:file*.
`IsHoldsLockFilter`	Filter nodes which do not hold lock.
`IsNotHoldsLockFilter`	Filter nodes which are holding lock.
`IsNotRootNodeFilter`	Filter the root node.
`IsInTrashFilter`	Filter nodes that are not in the trash node.
`IsNotInTrashFilter`	Filter nodes that are in the trash node.
`IsNotSameNameSiblingFilter`	Filter nodes that allow the same name siblings.
`IsMixCommentable`	Filter nodes that do not allow commenting.
`IsMixVotable`	Filter nodes that do not allow voting.
`IsNotSimpleLockedFilter`	Filter nodes that are locked.
`IsNotSymlinkFilter`	Filter nodes that are symlinks.
`IsNotCategoryFilter`	Filter nodes that are of the category type.

Filters	Description
`IsNotSystemWorkspaceFilter`	Filter actions of the system-typed workspace.
`IsNotCheckedOutFilter`	Filter nodes that are checked out.
`IsTrashHomeNodeFilter`	Filter nodes that are not trash ones.
`IsNotTrashHomeNodeFilter`	Filter a node that is the trash one.
`IsNotEditingDocumentFilter`	Filter nodes that are being edited.
`IsPasteableFilter`	Filter nodes where the **paste** action is not allowed.
`IsReferenceableNodeFilter`	Filter nodes that do not allow adding references.
`IsNotFolderFilter`	Filter nodes that are folders.
`IsCheckedOutFilter`	Filter nodes that are not checked out.
`IsVersionableFilter`	Filter nodes which do not allow versioning.
`IsVersionableOrAncestorFilter`	Filter nodes and ancestor nodes which do not allow versioning.
`IsDocumentFilter`	Filter nodes that are not documents.
`IsEditableFilter`	Filter nodes that are not editable.

4.3.2.4.2. Create your own filters

Beside using existing ones, you can also create new filters. See the example code below:

```
public class MyUIFilter implements UIExtensionFilter {

/**
* This method checks if the current node is of the right type
*/
public boolean accept(Map<String, Object> context) throws Exception {
// Retrieve the current node from the context
   Node currentNode = (Node) context.get(Node.class.getName());
return currentNode.isNodeType("exo:article");
}

/**
* This is the type of the filter
*/
public UIExtensionFilterType getType() {
return UIExtensionFilterType.MANDATORY;
}

/**
* This is called when the filter has failed
*/
public void onDeny(Map<String, Object> context) throws Exception {
System.out.println("This document has been rejected");
}
}
```

4.3.2.5. Other toolbars

Working with other toolbars is quite similar to *UIActionbar*, except configurations and resources.

Side bar

- **Sample configuration**

```
<object-param>
 <name>Example</name>
```

```
 <object type="org.exoplatform.webui.ext.UIExtension">
  <field name="type"><string>org.exoplatform.ecm.dms.UISideBar</string></field>
  <field name="name"><string>Example</string></field>
  <field name="rank"><int>110</int></field>
  <field name="component"><string>com.acme.ExampleActionComponent</string></field>
 </object>
</object-param>
```

Resources are located at *$TOMCAT-HOME/webapps/ecmexplorer/WEB-INF/classes/locale/portlet/explorer/JCRExplorerPortlet_en.xml* (for English which is also the default language):

```
...
 <UISideBar>
   ...
  <label>
    <example>Example action</example>
   ...
  </label>
   ...
 </UISideBar>
...
```

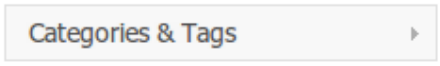
Admin control panel

- **Sample configuration**

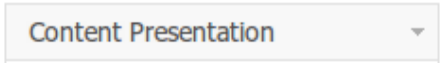
```
<object-param>
 <name>Example</name>
 <object type="org.exoplatform.webui.ext.UIExtension">
  <field name="type"><string>org.exoplatform.ecm.dms.UIECMAdminControlPanel</string></field>
  <field name="rank"><int>110</int></field>
  <field name="name"><string>Example</string></field>
  <field name="category"><string>Ontologies</string></field>
  <field name="component"><string>com.acme.ExampleActionComponent</string></field>
 </object>
</object-param>
```

The "category" field specifies the category where your extension action is performed. There are 4 options:

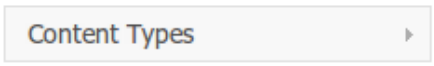
- Ontology



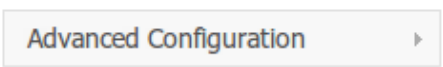
- ContentPresentation



- Content Type



- Advanced Configuration



Resources are located at *$TOMCAT-HOME/webapps/ecmadmin/WEB-INF/classes/locale/portlet/administration/ECMAdminPortlet_en.xml* (for English which is also the default language):

```
...
 <UIECMAdminControlPanel>
   ...
  <label>
    <example>Example panel</example>
   ...
  </label>
   ...
 </UIECMAdminControlPanel>
...
```

Context menu

- **Sample configuration**

```
<object-param>
 <name>Example</name>
 <object type="org.exoplatform.webui.ext.UIExtension">
  <field name="type"><string>org.exoplatform.ecm.dms.UIWorkingArea</string></field>
  <field name="rank"><int>105</int></field>
  <field name="name"><string>Example</string></field>
  <field name="category"><string>ItemContextMenu_SingleSelection</string></field>
  <field name="component"><string>com.acme.ExampleActionComponent</string></field>
 </object>
</object-param>
```

The "category" field specifies the category where your extension action is performed. There are many options:

- `ItemContextMenu_SingleSelection`: This menu has only one item when Trash Folder is right-clicked.
- `ItemContextMenu`: The menu appears when the user selects one or many items.
- `GroundContextMenu` & `ItemGroundContextMenu`: The menu appears when the user right-clicks the ground of node.

Resources are located at *$TOMCAT-HOME/webapps/ecmexplorer/WEB-INF/classes/locale/portlet/explorer/JCRExplorerPortlet_en.xml* (for English which is also the default language):

```
<UIWorkingArea>
   ...
  <label>
    <example>Example action</example>
   ...
  </label>
   ...
</UIWorkingArea>
```

File Viewer

- **Sample configuration**

```
<object-param>
 <name>Example</name>
 <object type="org.exoplatform.webui.ext.UIExtension">
  <field name="type"><string>org.exoplatform.ecm.dms.FileViewer</string></field>
  <field name="rank"><int>100</int></field>
  <field name="name"><string>Example</string></field>
```

```
<field name="category"><string>FileViewer</string></field>
<field name="component"><string>com.acme.ExampleActionComponent</string></field>
<field name="extendedFilters">
  <collection type="java.util.ArrayList">
    <value>
      <object type="org.exoplatform.webui.ext.filter.impl.FileFilter">
        <field name="mimeTypes">
          <collection type="java.util.ArrayList">
            <value><string>foo/bar</string></value>
          </collection>
        </field>
      </object>
    </value>
  </collection>
 </field>
</object>
</object-param>
```

Resources are located at *$TOMCAT-HOME/webapps/ecm-wcm-extension/WEB-INF/classes/locale/ecm/views_en.xml* (for English which is also the default language):

```
<File>
  <view>
    <example>Example view</example>
   ...
  </view>
 </File>
```

4.3.3. Authoring Extension

- **Extended Publication Plugin**

  Details of an extended Publication plugin used to manage the lifecycles of documents in Content, including:

  - **States [136]**

    Information about new states and new profiles of the extended publication that are enabled in Content.

  - **Start/End publication dates [136]**

    Introduction to new properties added to the new publication plugin that allows you to manage the content publication in a defined period.

  - **New Publication Mixin [136]**

    Introduction to the new authoring mixin that supplies more information about the document creator.

- **Publication Manager**

  Introduction to Publication Manager which manages lifecycles and contexts in Content and its details, including:

  - **Lifecycle**

    Sample code of lifecycle, information about 3 lifecycles, and instructions on how to listen to a lifecycle and to perform tasks when a content's state is updated.

  - **Context**

    Details of context, its sample code and rules.

  - **New Authoring Mixin**

    Introduction to the new authoring mixin that supplies more information about the document creator, its sample code and details of querying based on publication status.

4.3.3.1. Extended Publication Plugin

This section covers the following topics:

States

This extended publication has new states and new profiles that are enabled in 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 i18 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.

Start/End publication datess

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 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).

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" onParentVersion="IGNORE" protected="false"
 requiredType="Date">
      <valueConstraints/>
    </propertyDefinition>
    <propertyDefinition autoCreated="false" mandatory="true" multiple="false" name="publication:endPublishedDate" onParentVersion="IGNORE" protected="false"
 requiredType="Date">
      <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 Content platform. It allows managing 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>lifecyles</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.LifecyclesConfig$Lifecycle">
                  <field name="name">
                    <string>lifecycle1</string>
                  </field>
                  <field name="publicationPlugin">
                    <string>States and versions based publication</string>
                  </field>
                </object>
              </value>
              <value>
                <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.LifecyclesConfig$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.LifecyclesConfig$State">
                          <field name="state">
                            <string>draft</string>
                          </field>
                          <field name="memberships">
                            <collection type="java.util.ArrayList">
                              <value>
                                <string>author:/communication</string>
                              </value>
                              <value>
                                <string>author:/sanitaryAlert</string>
                              </value>
                              <value>
                                <string>author:/informations</string>
                              </value>
                            </collection>
                          </field>
                        </object>
                      </value>
                      <value>
                        <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.LifecyclesConfig$State">
                          <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.LifecyclesConfig$State">
```

```
                        <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.LifecyclesConfig$State">
                        <field name="state">
                          <string>staged</string>
                        </field>
                        <field name="membership">
                          <string>publisher:/platform/web-contributors</string>
                        </field>
                      </object>
                    </value>
                    <value>
                      <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.LifecyclesConfig$State">
                        <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.LifecyclesConfig$Lifecycle">
                <field name="name">
                  <string>lifecycle3</string>
                </field>
                <field name="publicationPlugin">
                  <string>Authoring publication</string>
                </field>
                <field name="states">
                  <collection type="java.util.ArrayList">
                    <value>
                      <object type="org.exoplatform.services.wcm.publication.lifecycles.impl.LifecyclesConfig$State">
                        <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.LifecyclesConfig$State">
                        <field name="state">
                          <string>published</string>
                        </field>
                        <field name="memberships">
                          <collection type="java.util.ArrayList">
                            <value>
                              <string>publisher:/communication</string>
                            </value>
                            <value>
                              <string>publisher:/sanitaryAlert</string>
                            </value>
                            <value>
                              <string>publisher:/informations</string>
                            </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 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>
```

4.3.3.2.1.2. Perform tasks when a content's state is updated

To perform some tasks when a content's state is updated, you need to create a listener that handles the task and configure it. Following is the general configuration:

```
<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>my.package.MyListener</type>
```

```
<description>Your listener description</description>
</component-plugin>
</external-component-plugins>
```

With this configuration, your listener *my.package.MyListener* will be executed each time a content's state is changed.

For example, eXo provides a listener which automatically sends email notifications about the new state to all users of defined groups: *org.exoplatform.wcm.authoring.listener.PostUpdateStateEventListener*. So, the configuration will be:

```
<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.wcm.authoring.listener.PostUpdateStateEventListener</type>
<description>This listener will send a mail when there are changes in a content's state</description>
</component-plugin>
</external-component-plugins>
```

4.3.3.2.2. Context

A context is defined by simple rules. In 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$Context">
                  <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$Context">
    <field name="name">
      <string>context1</string>
    </field>
    <field name="priority">
      <string>100</string>
    </field>
    <field name="lifecycle">
      <string>lifecycle1</string>
    </field>
    <field name="membership">
      <string>*:/platform/web-contributors</string>
    </field>
    <field name="site">
      <string>acme</string>
    </field>
    <field name="path">
      <string>repository:collaboration:/sites content/live/acme/categories</string>
    </field>
  </object>
  <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig$Context">
    <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$Context">
    <field name="name">
      <string>context3</string>
    </field>
    <field name="priority">
      <string>80</string>
    </field>
    <field name="lifecycle">
      <string>lifecycle3</string>
    </field>
    <field name="membership">
      <string>manager:/company/finances</string>
    </field>
    <field name="path">
      <string>repository:collaboration:/documents/company/finances</string>
    </field>
  </object>
  <object type="org.exoplatform.services.wcm.publication.context.impl.ContextConfig$Context">
    <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:/communication</string>
      </value>
      <value>
        <string>manager:/sanitaryAlert</string>
      </value>
      <value>
```

```
                        <string>manager:/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 lifecyle 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" onParentVersion="IGNORE" protected="false"
 requiredType="String">
    <valueConstraints/>
  </propertyDefinition>
        <propertyDefinition autoCreated="false" mandatory="false" multiple="false" name="publication:lifecycle" onParentVersion="IGNORE" protected="false"
 requiredType="String">
    <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".

4.3.4. Auxiliary attributes for documents

By default, your activities, such as writing a document, and uploading a file, are published on the activity stream. However, you can decide to publish these activities or not by creating a context named *DocumentContext* for a specific document. This context stores some auxiliary attributes of the document and helps document listeners make decision based on these attributes.

This context looks like:

```
public class DocumentContext {
private static ThreadLocal<DocumentContext> current = new ThreadLocal<DocumentContext>();

public static DocumentContext getCurrent() {
  if (current.get() == null) {
    setCurrent(new DocumentContext());
   }
  return current.get();
 }

....
//Each time, attributes are able to set and got via:
/**
 * @return the attributes
 */
 public HashMap<String, Object> getAttributes() {
  return attributes;
 }

 /**
 * @param attributes the attributes to set
 */
 public void setAttributes(HashMap<String, Object> attributes) {
   this.attributes = attributes;
 }
}
```

For example:

When you upload a document to a drive by using *ManageDocumentService*, but do not want to publish this activity on the activity stream, you can do as follows:

```
DocumentContext.getCurrent().getAttributes().put(DocumentContext.IS_SKIP_RAISE_ACT, true);
```

Then, this activity is skipped at:

```
Object isSkipRaiseAct = DocumentContext.getCurrent().getAttributes().get(DocumentContext.IS_SKIP_RAISE_ACT);
if (isSkipRaiseAct != null && Boolean.valueOf(isSkipRaiseAct.toString())) {
 return;
}
```

Note

The *DocumentContext* class is able to help developers manage various kinds of actions with a document based on its auxiliary attributes. You can be free to define new attributes for yourself.

4.4. CMIS Usage code examples

- **Login to repository**

  Example of using Java to login to repository.
- **List of documents (folder, files)**

  Description about the usage of several methods to get the documents lists, such as *getChildren()*, *getFolderTree()* and *getDescentants()*.
- **Read document properties and content-stream**

  Instructions on how to read and get the document properties and content stream.
- **Search of data and syntax examples**

  Examples of using Java and Javascript to search for data and syntax in CMIS.
- **Modification of document properties or content**

  Instructions on how to use Java and Javascript to update and get document properties or content in CMIS.

The examples of the CMIS usage may be useful for developers who need to access a repository. CMIS access code snippets are built using Apache HTTP Client for Java, or using Google gadgets (gadgets.io) for JavaScript examples. For examples of CURL, visit http://code.google.com/p/xcmis/wiki/xCMISusesWithCurl.

See also

- WCM Templates
- WCM Explorer
- Extensions
- Public REST APIs
- Public Java APIs

- Deprecated portlets
- Miscellaneous and Tips
- FAQs

4.4.1. Login to repository

Note

The CMIS service uses the default authentication in general case, but it can be overridden in case of embedding CMIS into an Application Service. In these examples, only the Basic HTTP authentication is covered.

Use java

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.GetMethod;

HttpClient client = new HttpClient();
client.getState().setCredentials(
    new AuthScope("localhost", 8080, "realm"),
    new UsernamePasswordCredentials("root", "exo");
....
```

4.4.2. List of documents (folder, files)

There are several methods to get the documents lists, such as getChildren(), getFolderTree() and getDescentants(), their usage will be described below. The difference between them is the usage of different URL segments to get data ("/children" for getChildren(), "/foldertree" for getFolderTree(), "/descendants" for getDescentants()), and a different kind of results (getChildren() returns a flat structure, while a getFolderTree() and getDescentants() have a tree of items in response).

Use Java

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.MultiThreadedHttpConnectionManager;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/children/";
url += obj_id;

HttpClient client = new HttpClient(new MultiThreadedHttpConnectionManager());
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

GetMethod get = new GetMethod(url);
try {
int result = client.executeMethod(get);
final String strResponse = get.getResponseBodyAsString();
} finally {
  get.releaseConnection();
}
```

Use JavaScript

Creating an URL to make a request (consists of repository name, the method name, for example "/children/", and folderID to get children from):

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/children/";
url += obj_id;
```

Performing request:

```
var params = {};
params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.GET;
params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.ContentType.FEED;
gadgets.io.makeRequest(url, handler, params);
```

Processing results (the code is located in the handler is specified while making a request, the same way it might be used for all examples in this chapter):

```
var handler = function(resp) {
  var data = eval(resp.data.Entry);
   for (var i = 0; i < data.length; i++) {
     var doc = data[i];
     alert(doc.Title);
     alert(doc.Date);
      ...etc..
   }
}
```

4.4.3. Read document properties and content-stream

Reading the Document properties and content stream are two separate operations. Getting the content stream is possible after the properties set have been read and the content stream ID was extracted from it.

Use Java

Get document properties.

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.MultiThreadedHttpConnectionManager;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/object/";
url += obj_id;

HttpClient client = new HttpClient(new MultiThreadedHttpConnectionManager());
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

GetMethod get = new GetMethod(url);
try {
  int result = client.executeMethod(get);
  final String strResponse = get.getResponseBodyAsString();
  // use response...
} finally {
```

```
  get.releaseConnection();
}
```

Get document content-stream.

To get the Document's content stream, an URL must contain "/file" part, object ID, and optionally the content stream ID, which can be used, for example, to obtain renditions. If no stream ID is specified, the default stream will be returned.

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.PostMethod;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/file/";
url += obj_id;
//Optionally
url += "?";
url += "streamid=";
url += streamID;

HttpClient client = new HttpClient();
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

GetMethod get = new GetMethod(url);
try {
  int result = client.executeMethod(get);
  final InputStream stream = get.getResponseBodyAsStream();
  try {
  // use stream...
  int dataByte = stream.read();
  } finally {
    stream.close();
  }
} finally {
  get.releaseConnection();
}
```

Use JavaScrip

Get document properties.

Creating an URL to make a request (consists of repository name, method name, for example "/children/", and folder ID to get the children from):

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/object/";
url += obj_id;
```

Performing request:

```
var params = {};
params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.GET;
params[gadgets.io.RequestParameters.CONTENT_TYPE] = gadgets.io.ContentType.FEED;
gadgets.io.makeRequest(url, handler, params);
```

You can also use the ContentType.DOM parameter to parse the feed in your application (Using DOMParser for example).

Get document content-stream.

Note

Performing a content stream request in JavaScript will cause the browser dialog for a file download.

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/file/";
url += obj_id;
//Optionally
url += "?";
url += "streamid=";
url += streamID;
```

4.4.4. Search of data and syntax examples

CMIS supports SQL queries for more handful content search. Query service can handle both GET and POST requests. URL for query consists of repository name and method name "/query". The GET request must contain query as a parameter named "q", in case of POST request query must be located in a request body.

For more detailed instructions how to construct queries, refer to the Query examples chapter.

Use Java

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.StringRequestEntity;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/query/";

HttpClient client = new HttpClient();
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

PostMethod post = new PostMethod(url);
String s = "<?xml version='1.0' encoding='utf-8'?>"
      + "<cmis:query xmlns='http://www.w3.org/2005/Atom' xmlns:cmis='http://docs.oasis-open.org/ns/cmis/core/200908/'>"
      + "<cmis:statement>SELECT * FROM cmis:document</cmis:statement>"
      + "<cmis:maxItems>10</cmis:maxItems>"
      + "<cmis:skipCount>0</cmis:skipCount>"
      + "<cmis:searchAllVersions>true</cmis:searchAllVersions>"
      + "<cmis:includeAllowableActions>true</cmis:includeAllowableActions>"
      + "</cmis:query>";

RequestEntity entity = new StringRequestEntity(s, "text/xml","utf-8");
try {
  post.setRequestEntity(entity);
  int result = client.executeMethod(post);
  final String strResponse = post.getResponseBodyAsString();
  // use response...
} finally {
  post.releaseConnection();
}
```

Use JavaScript

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/query/";
```

```
var params = {};
params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.POST;
params[gadgets.io.RequestParameters.POST_DATA] = gadgets.io.encodeValues(someQuery);
gadgets.io.makeRequest(url, handler, params);
```

4.4.5. Modification of document properties or content

The command of property update uses PUT method. The URL is the same as the one for reading properties, the difference is only in the HTTP method used. The body of the request must be an Atom document with specified properties (see spec. 2.2.4.12 for detailed constructing document).

Sending of content stream can be executed via PUT or POST requests. Content-type of the request must be an "multipart/form-data".

Use Java

Update properties:

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.StringRequestEntity;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.RequestEntity;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/object/";
url += obj_id;

HttpClient client = new HttpClient();
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

String atomDoc = "<?xml version='1.0' encoding='utf-8'?>"
        + "<entry xmlns='http://www.w3.org/2005/Atom'"
        + " xmlns:cmis='http://docs.oasis-open.org/ns/cmis/core/200908/'"
        + " xmlns:cmisra='http://docs.oasis-open.org/ns/cmis/restatom/200908/'>"
        + "<cmisra:object><cmis:properties>"
        + "<cmis:propertyString queryName='cmis:name' localName='cmis:name' propertyDefinitionId='cmis:name'>"
        + "<cmis:value>newName</cmis:value>"
        + "</cmis:propertyString>"
        + "</cmis:properties></cmisra:object>"
        + "</entry>";

PutMethod put = new PutMethod(url);
RequestEntity entity = new StringRequestEntity(atomDoc, "text/xml", "utf-8");
put.setRequestEntity(entity);

try {
  int result = client.executeMethod(put);
  final String strResponse = put.getResponseBodyAsString();
} finally {
  put.releaseConnection();
}
```

Set content stream:

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.InputStreamRequestEntity;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.RequestEntity;

String url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/file/";
url += obj_id;

HttpClient client = new HttpClient();
client.getHttpConnectionManager().
getParams().setConnectionTimeout(10000);

PostMethod post = new PostMethod(url);
RequestEntity entity = new InputStreamRequestEntity(inputStream, "text/xml; charset=ISO-8859-1");
post.setRequestEntity(entity);

try {
  int result = client.executeMethod(post);
  final String strResponse = post.getResponseBodyAsString();
} finally {
  post.releaseConnection();
}
```

Use JavaScript

Update properties:

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/object/";
url += obj_id;
```

```
//constructing document
 String atomDoc = "<?xml version='1.0' encoding='utf-8'?>";
      atomDoc += "<entry xmlns='http://www.w3.org/2005/Atom'";
      atomDoc += " xmlns:cmis='http://docs.oasis-open.org/ns/cmis/core/200908/'";
      atomDoc += " xmlns:cmisra='http://docs.oasis-open.org/ns/cmis/restatom/200908/'>";
      atomDoc += "<cmisra:object><cmis:properties>";
      atomDoc += "<cmis:propertyString queryName='cmis:name' localName='cmis:name' propertyDefinitionId='cmis:name'>";
      atomDoc += "<cmis:value>newName</cmis:value>";
      atomDoc += "</cmis:propertyString>";
      atomDoc += "</cmis:properties></cmisra:object></entry>";

 var params = {};
 params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.PUT;
 params[gadgets.io.RequestParameters.POST_DATA] = atomDoc;
 gadgets.io.makeRequest(url, handler, params);
```

Set content stream:

```
var url = "http://localhost:8080/rest/private/cmisatom/";
url += repository;
url += "/file/";
url += obj_id;
```

```
var params = {};
params[gadgets.io.RequestParameters.METHOD] = gadgets.io.MethodType.POST;
params[gadgets.io.RequestParameters.CONTENT_TYPE] = "multipart/form-data";
params[gadgets.io.RequestParameters.POST_DATA] = contentStream;
gadgets.io.makeRequest(url, handler, params);
```

4.5. Public REST APIs

- **ThumbnailRESTService**

  Return a responding data as a thumbnail image.
- **RssConnector**

  Generate an RSS feed.
- **FCKCoreRESTConnector**

  Get a list of files and folders, and create a folder and upload files.
- **ResourceBundleConnector**

  Get the bundle basing on the key and the locale.
- **VoteConnector**

  Return and set a vote value of a given node in the sent parameter.
- **DriverConnector**

  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.
- **GadgetConnector**

  Instantiate a new gadget connector.
- **PortalLinkConnector**

  Return a page URI for a given location.
- **GetEditedDocumentRESTService**

  Return the latest edited documents.
- **PublicationGetDocumentRESTService**

  Return a list of published documents.
- **FavoriteRESTService**

  Return a list of favorite documents of a given user.
- **RESTImagesRendererService**

  Get the image binary data of a given image node.
- **LifecycleConnector**

  Return a list of contents in a given state range of the publication lifecycle.
- **CopyContentFile**

  Copy a file.
- **PDFViewerRESTService**

  Return the PDF content to display on the web page
- **ManageDocumentService**

Allow performing some actions on a folder or a file, such as creating, deleting a folder/file, or uploading a file.

- **DownloadConnector**

  Enable downloading the content of *nt:file*.

See also

- WCM Templates
- WCM Explorer
- Extensions
- CMIS Usage code examples
- Public Java APIs
- Deprecated portlets
- Miscellaneous and Tips
- FAQs

4.5.1. ThumbnailRESTService

Resource	Description
GET /medium/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at a medium size (64x64). For example: /portal/rest/thumbnailImage/medium/repository/collaboration/test.gif/
GET /big/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at a big size.
GET /large/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at a large size (300x300).
GET /small/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at a small size (32x32).
GET /custom/{size}/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at a custom size.
GET /origin/{repoName}/{workspaceName}/{nodePath:.*}/	Return an image at an original size.

4.5.1.1. GET /medium/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at a medium size (64x64). For example: /portal/rest/thumbnailImage/medium/repository/collaboration/test.gif/

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/medium/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.1.2. GET /big/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at a big size.

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/big/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.1.3. GET /large/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at a large size (300x300).

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/large/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.1.4. GET /small/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at a small size (32x32).

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/small/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.1.5. GET /custom/{size}/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at a custom size.

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/custom/{size}/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
size	The customized size of the image.
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.1.6. GET /origin/{repoName}/{workspaceName}/{nodePath:.*}/

Return an image at an original size.

URL:

```
http://{domain_name}/{rest_context_name}/private/thumbnailImage/origin/{repoName}/{workspaceName}/{nodePath:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
nodePath	The node path.

- **Optional (query parameters)**: No

4.5.2. RssConnector

Resource	Description
GET /rss/	Generate an RSS feed.

4.5.2.1. GET /rss/

Generate an RSS feed.

URL:

```
http://{domain_name}/{rest_context_name}/private/feed/rss/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repository	The name of repository.
workspace	The name of workspace.
server	The server.
siteName	The name of site.
title	The title of the feed.
desc	The description of the feed.
folderPath	The folder path of the feed.
orderBy	The criteria to order the content.
orderType	The descending or ascending order.
lang	The language of the feed.
detailPage	The page used to open the content.
detailParam	The parameters is the key in the URL to let CLV know which really is the path in the current URL.
recursive	This param is deprecated and will be moved soon.

4.5.3. FCKCoreRESTConnector

Resource	Description
GET /getFoldersAndFiles/	Return folders and files in the current folder.
GET /createFolder/	Create a folder under the current folder.
POST /uploadFile/upload/	Upload a file with the HttpServletRequest.
GET /uploadFile/control/	Control the process of uploading a file, such as aborting, deleting or progressing the file.

4.5.3.1. GET /getFoldersAndFiles/

Return folders and files in the current folder.

URL:

```
http://{domain_name}/{rest_context_name}/private/fckconnector/jcr/getFoldersAndFiles/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
currentFolder	The current folder.
command	The command.
type	The type.

4.5.3.2. GET /createFolder/

Create a folder under the current folder.

URL:

```
http://{domain_name}/{rest_context_name}/private/fckconnector/jcr/createFolder/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
currentFolder	The current folder.
newFolderName	The name of the new folder.
language	The language.

4.5.3.3. POST /uploadFile/upload/

Upload a file with the HttpServletRequest.

URL:

```
http://{domain_name}/{rest_context_name}/private/fckconnector/jcr/uploadFile/upload/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**: No

4.5.3.4. GET /uploadFile/control/

Control the process of uploading a file, such as aborting, deleting or progressing the file.

URL:

```
http://{domain_name}/{rest_context_name}/private/fckconnector/jcr/uploadFile/control/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The repository name.
workspaceName	The workspace name.
currentFolder	The current folder.
action	The process of the upload file, such as saving or cancelling the file.
language	The language of the file.
fileName	The file name.
uploadId	The Id of the upload.

4.5.4. ResourceBundleConnector

Resource	Description
GET /getBundle/	Get the bundle that is based on the key and the locale.

4.5.4.1. GET /getBundle/

Get the bundle that is based on the key and the locale.

URL:

```
http://{domain_name}/{rest_context_name}/private/bundle/getBundle/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
key	The key used to get the bundle.
locale	The locale used to get the bundle.

4.5.5. VoteConnector

Resource	Description
POST /star/	Set a vote value for a given content.
GET /star/	Return a vote value for a given content.
GET /postVote/	Set a vote value for a given content.
GET /getVote/	Return a vote value for a given content.

4.5.5.1. POST /star/

Set a vote value for a given content.

URL:

```
http://{domain_name}/{rest_context_name}/private/contents/vote/star/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**: No

4.5.5.2. GET /star/

Return a vote value for a given content.

URL:

```
http://{domain_name}/{rest_context_name}/private/contents/vote/star/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
jcrPath	The path of the content.

4.5.5.3. GET /postVote/

Set a vote value for a given content.

URL:

```
http://{domain_name}/{rest_context_name}/private/contents/vote/postVote/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
jcrPath	The path of the content.
vote	The vote value.
lang	The language of the content.

4.5.5.4. GET /getVote/

Return a vote value for a given content.

URL:

```
http://{domain_name}/{rest_context_name}/private/contents/vote/getVote/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
jcrPath	The path of the content.

4.5.6. DriverConnector

Resource	Description
GET /getDrivers/	Return a list of drives for the current user.
GET /getFoldersAndFiles/	Return all folders and files in a given location.
POST /uploadFile/upload/	Upload a file.
GET /uploadFile/control/	Control the process of uploading a file, such as aborting, deleting or progressing the file.

4.5.6.1. GET /getDrivers/

Return a list of drives for the current user.

URL:

```
http://{domain_name}/{rest_context_name}/private/wcmDriver/getDrivers/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
lang	The language of the drive name.

4.5.6.2. GET /getFoldersAndFiles/

Return all folders and files in a given location.

URL:

```
http://{domain_name}/{rest_context_name}/private/wcmDriver/getFoldersAndFiles/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
driverName	The name of drive.
currentFolder	The current folder.
currentPortal	The current portal.
repositoryName	The name of repository.
workspaceName	The name of workspace.
filterBy	The type of filter.

4.5.6.3. POST /uploadFile/upload/

Upload a file.

URL:

```
http://{domain_name}/{rest_context_name}/private/wcmDriver/uploadFile/upload/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
uploadId	The Id of upload.

4.5.6.4. GET /uploadFile/control/

Control the process of uploading a file, such as aborting, deleting or progressing the file.

URL:

```
http://{domain_name}/{rest_context_name}/private/wcmDriver/uploadFile/control/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
repositoryName	The name of repository.
workspaceName	The name of workspace.
driverName	The name of drive.
currentFolder	The current folder.
currentPortal	The current portal.
userId	The user identity.
jcrPath	The path of the file.
action	The action.
language	The language.
fileName	The name of file.
uploadId	The Id of upload.

4.5.7. GadgetConnector

Resource	Description
GET /getFoldersAndFiles/	Get folders and files.

4.5.7.1. GET /getFoldersAndFiles/

Get folders and files.

URL:

```
http://{domain_name}/{rest_context_name}/private/wcmGadget/getFoldersAndFiles/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
currentFolder	The current folder.
lang	The language.
host	The server address on which the gadget is deployed.

4.5.8. PortalLinkConnector

Resource	Description
GET /getFoldersAndFiles/	Get the page URI.

4.5.8.1. GET /getFoldersAndFiles/

Get the page URI.

URL:

```
http://{domain_name}/{rest_context_name}/private/portalLinks/getFoldersAndFiles/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
currentFolder	The current folder.
command	The command.
type	The type.

4.5.9. GetEditedDocumentRESTService

Resource	Description
GET /{repository}/	Return the latest edited documents.

4.5.9.1. GET /{repository}/

Return the latest edited documents.

URL:

```
http://{domain_name}/{rest_context_name}/private/presentation/document/edit/{repository}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repository	The name of repository.

- **Optional (query parameters)**:

Parameter	Description
showItems	Return the number of items in each page.

Parameter	Description
showGadgetWs	Show the gadget workspace or not.

4.5.10. PublicationGetDocumentRESTService

Resource	Description
GET /{repository}/{workspace}/{state}/	Return a list of published documents by the default plugin. For example: /portal/rest/publication/presentation/{repository}/{workspace}/{state}?showItems={numberOfItem}
GET /{repository}/{workspace}/{publicationPluginName}/{state}/	Return a list of published documents by a specific plugin. For example: /portal/rest/publication/presentation/{repository}/{workspace}/{publicationPluginName}/{state}?showItems={numberOfItem}

4.5.10.1. GET /{repository}/{workspace}/{state}/

Return a list of published documents by the default plugin. For example: /portal/rest/publication/presentation/{repository}/{workspace}/{state}?showItems={numberOfItem}

URL:

```
http://{domain_name}/{rest_context_name}/private/publication/presentation/{repository}/{workspace}/{state}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repository	The name of repository.
workspace	The name of workspace.
state	The state is specified to classify the process.

- **Optional (query parameters)**:

Parameter	Description
showItems	Show the number of items per page.

4.5.10.2. GET /{repository}/{workspace}/{publicationPluginName}/{state}/

Return a list of published documents by a specific plugin. For example: /portal/rest/publication/presentation/{repository}/{workspace}/{publicationPluginName}/{state}?showItems={numberOfItem}

URL:

```
http://{domain_name}/{rest_context_name}/private/publication/presentation/{repository}/{workspace}/{publicationPluginName}/{state}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repository	The repository name.
workspace	The workspace name.
publicationPluginName	The name of the plugin.
state	The state is specified to classify the process.

- **Optional (query parameters)**:

Parameter	Description
showItems	Show the number of items per page.

4.5.11. FavoriteRESTService

Resource	Description
GET /all/{repoName}/{workspaceName}/{userName}	Return a list of favorite documents of a given user.

4.5.11.1. GET /all/{repoName}/{workspaceName}/{userName}

Return a list of favorite documents of a given user.

URL:

```
http://{domain_name}/{rest_context_name}/private/favorite/all/{repoName}/{workspaceName}/{userName}
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
userName	The username.

- **Optional (query parameters)**:

Parameter	Description
showItems	Show the number of items per page.

4.5.12. RESTImagesRendererService

Resource	Description
GET /{repositoryName}/{workspaceName}/{nodeIdentifier}	Get the image binary data of a given image node.

4.5.12.1. GET /{repositoryName}/{workspaceName}/{nodeIdentifier}

Get the image binary data of a given image node.

URL:

```
http://{domain_name}/{rest_context_name}/private/images/{repositoryName}/{workspaceName}/{nodeIdentifier}
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
repositoryName	The repository name.
workspaceName	The workspace name.
nodeIdentifier	The node identifier.

- **Optional (query parameters)**:

Parameter	Description
param	Check if the document is a file or not.

4.5.13. LifecycleConnector

Resource	Description
GET /bystate/	Return a list of content from a given to the last state. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/bystate/?fromstate =draft&user=root&lang=en&workspace=collaboration
GET /tostate/	Return a list of content from the beginning to the last state. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/tostate/?fromstate =draft&tostate=pending&user=root&lang=en&workspace=collaboration
GET /bydate/	Return a list of content from the given beginning to published state and before the given date. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/bydate/?fromstate =staged&date=2&lang=en&workspace=collaboration

4.5.13.1. GET /bystate/

Return a list of content from a given to the last state. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/bystate/?fromstate =draft&user=root&lang=en&workspace=collaboration

URL:

```
http://{domain_name}/{rest_context_name}/private/authoring/bystate/
```

Parameters:

- **Required (path parameters)**: No

- **Optional (query parameters)**:

Parameter	Description
fromstate	The beginning state of the content.
user	The author of the content.
lang	The language of the content.
workspace	The workspace name which contains the content.
json	The format of the returned data.

4.5.13.2. GET /tostate/

Return a list of content from the beginning to the last state. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/tostate/?fromstate =draft&tostate=pending&user=root&lang=en&workspace=collaboration

URL:

```
http://{domain_name}/{rest_context_name}/private/authoring/tostate/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
fromstate	The beginning state of the content.
tostate	The destination state of the content.
user	The author of the content.
lang	The language of the content.
workspace	The workspace name which contains the content.
json	The format of the returned data.

4.5.13.3. GET /bydate/

Return a list of content from the given beginning to published state and before the given date. For example: http://localhost:8080/ecmdemo/rest-ecmdemo/authoring/bydate/?fromstate =staged&date=2&lang=en&workspace=collaboration

URL:

```
http://{domain_name}/{rest_context_name}/private/authoring/bydate/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
fromstate	The beginning state of the content.

Parameter	Description
date	The date before when the content is published.
lang	The language of the content.
workspace	The workspace name which contains the content.
json	The format of the returned data.

4.5.14. CopyContentFile

Resource	Description
POST /copy/	Copy a file.

4.5.14.1. POST /copy/

Copy a file.

URL:

```
http://{domain_name}/{rest_context_name}/private/copyfile/copy/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**: No

4.5.15. PDFViewerRESTService

Resource	Description
GET	Return a thumbnail image for a PDF document.

4.5.15.1. GET

Return a thumbnail image for a PDF document.

Parameters:

- **Required (path parameters)**:

Parameter	Description
repoName	The name of repository.
workspaceName	The name of workspace.
uuid	The identifier of the document.
pageNumber	The page number.
rotation	The page rotation. The valid values are: 0.0f, 90.0f, 180.0f, 270.0f.
scale	The Zoom factor which is applied to the rendered page.

- **Optional (query parameters)**: No

4.5.16. ManageDocumentService

Resource	Description
GET /getDrives/	Get all drives by type (General, Group or Personal).
GET /getFoldersAndFiles/	Get all folders and files which can be viewed by the current user.
GET /deleteFolderOrFile/	Delete a folder/file.
GET /createFolder/	Create a new folder and return its information.
POST /uploadFile/upload/	Upload a file to the server.
GET /uploadFile/control/	Return information about the upload status of a file (upload percentage, file name, and more).

4.5.16.1. GET /getDrives/

Get all drives by type (General, Group or Personal).

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/getDrives/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
driveType	The types of drive (General, Group, or Personal).
showPrivate	Show the Private drive or not. The default value is false.
showPersonal	Show the Personal drive or not. The default value is false.

4.5.16.2. GET /getFoldersAndFiles/

Get all folders and files which can be viewed by the current user.

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/getFoldersAndFiles/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
driveName	The name of drive.
workspaceName	The name of workspace.
currentFolder	The path to the folder to achieve its folders and files.
showHidden	Show the hidden items or not. The default value is false.

4.5.16.3. GET /deleteFolderOrFile/

Delete a folder/file.

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/deleteFolderOrFile/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
driveName	The name of drive.
workspaceName	The name of workspace.
itemPath	The path to the folder/file.

4.5.16.4. GET /createFolder/

Create a new folder and return its information.

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/createFolder/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
driveName	The name of drive.
workspaceName	The name of workspace.
currentFolder	The path to the folder where a child folder is added .
folderName	The name of folder.

4.5.16.5. POST /uploadFile/upload/

Upload a file to the server.

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/uploadFile/upload/
```

Parameters:

- **Required (path parameters)**: No

- **Optional (query parameters)**:

Parameter	Description
uploadId	The Id of uploaded resource.

4.5.16.6. GET /uploadFile/control/

Return information about the upload status of a file (upload percentage, file name, and more).

URL:

```
http://{domain_name}/{rest_context_name}/private/managedocument/uploadFile/control/
```

Parameters:

- **Required (path parameters)**: No
- **Optional (query parameters)**:

Parameter	Description
workspaceName	The name of workspace.
driveName	The name of drive.
currentFolder	The path to the current folder .
currentPortal	The name of the current site.
action	The action to perform (saving, processing, and more).
language	The language of user.
fileName	The name of file.
uploadId	The Id of the uploaded resource.

4.5.17. DownloadConnector

Resource	Description
GET /download/{workspace}/{path:.*}/	Return to browser a stream got from *jcr:content/jcr:data* for downloading the content of the node.

4.5.17.1. GET /download/{workspace}/{path:.*}/

Return to browser a stream got from *jcr:content/jcr:data* for downloading the content of the node.

URL:

```
http://{domain_name}/{rest_context_name}/private/contents/download/{workspace}/{path:.*}/
```

Parameters:

- **Required (path parameters)**:

Parameter	Description
workspace	The workspace where to store the document node
path	The path to the document node

- **Optional (query parameters)**:

Parameter	Description
version	The version name

4.6. Public Java APIs

- **TaxonomyService**

  Include many functions which allow adding, finding, or deleting taxonomies from a node.
- **LinkManager**

  Provide APIs to work with the linked node or the link included in a node.
- **PublicationManager**

  Manage the content publication.
- **WCMComposer**

  Get content inside the WCM product.
- **NewFolksonomy**

  Manage all the tag and tag style. Currently it just supports adding/editing/removing Private and Public tags.
- **ApplicationTemplateManager**

  Manage dynamic groovy templates for WCM-based products.
- **NodeFinder**

  Find a node with a given path.
- **JodConverter**

  Convert documents into different office formats.
- **TimelineService**

  Get all documents by time frame (day, month, year).
- **SiteSearchService**

  Allow finding all information matching with a given keyword.
- **SEOService**

  Provide APIs to manage SEO data of a page or a content.
- **ManageViewService**

  Include many functions which allow adding, editing, deleting, and getting views.

See also

- WCM Templates
- WCM Explorer
- Extensions
- CMIS Usage code examples
- Public REST APIs
- Deprecated portlets
- Miscellaneous and Tips
- FAQs

4.6.1. 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 ReposioryException;	`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 session system or user session.	`List<Node>`	Return the list of all the root nodes of the taxonomy tree available.
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.
	`taxonomyName`: The name of the taxonomy to remove.	`void`	Remove the taxonomy tree definition.

Method	Param	Return	Description
removeTaxonomyTree(String taxonomyName) throws RepositoryException			
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(Strin repository, String workspace, String absPath) throws RepositoryException;	`repository`: The name of the repository `workspace`: The name of the workspace. `absPath`: The absolute path of the taxonomy node to remove.	`void`	Remove the taxonomy node located at the given absolute path.
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.

Method	Param	Return	Description
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. `taxonomyName`: The name of the taxonomy.	`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.
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.	`void`	Remove a category to the given node.

Method	Param	Return	Description
	`categoryPath`: The path of the category relative to the root node of the given 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. `categoryPath`: The path of the category relative to the given taxonomy.	`void`	Add a new category path to the given node.

Method	Param	Return	Description
	`system`: Check system provider or not.		
getTaxonomyTreeDefaultUserPermission();	N/A	Map<String, String[]>	Get the default permission for the user in taxonomy tree.
addTaxonomyPlugin(ComponentPlugin 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.

4.6.2. 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 exo:symlink, 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 exo:symlink, then add it to the parent node and return the link node.
createLink(Node parent, String linkType, Node target, String linkName) throws RepositoryException;	`parent`: The parent node of the link. `inkType`: The primary node type of the link must be a sub-type of exo:symlink, 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.

Method	Param	Return	Description
	`linkName`: The name of 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 exo:symlink. `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 exo:symlink.	`Node`	Get the target node of the given link using the user.
isTargetReachable(Node link) throws RepositoryException;	`link`: The node of type exo:symlink.	`boolean`	Check if the target node of the given link can be reached using the user session.
isTargetReachable(Node link, boolean system)throws RepositoryException;	`link`: The node of type exo:symlink. `system`	boolean	Check if the target node of the given link can be reached using the user session.
isLink(Item item) throws RepositoryException;	`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(link)throws RepositoryException;	`link`: The node of type exo:symlink.	`string`	Return the primary node type of the target.
getAllLinks(Node targetNode, String linkType, String repoName)throws Exception	`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. (Deprecated)

Method	Param	Return	Description
getAllLinks(Node targetNode, String linkType, SessionProvider sessionProvider) throws Exception;	`targetNode`: The target node to get links. `linkType`: The type of the link to get. `sessionProvider`: The session provider	`List<Node>` - the list of link of the target node with given type.	Return all links of the given node.

4.6.3. PublicationManager

PublicationService is to manage the publication.

Method	Param	Return	Description
addLifecycle(ComponentPlug plugin)	`plugin`	`void`	Add publication plugin to the publication service.
removeLifecycle(Component plugin)	`plugin`	`void`	Remove publication plugin from the publication service.
addContext(ComponentPlugir plugin)	`plugin`	`void`	Add publication plugin context to the publication service.
removeContext(ComponentP plugin)	`plugin`	`void`	Remove publication plugin context from 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 workspace) throws Exception;	`fromstate/tostate` - The current range state of node. `user` - The last user of node. `lang` - The node's language. `workspace` - The Workspace of the node's location.	`List<Node>` `List<Lifecycle>`	Get all the nodes.

Method	Param	Return	Description
getLifecyclesFromUser(String remoteUser, String state);	`remoteUser` - The current user of publication service. `state` - The current state of the node.		Get all the Lifecycle of a specific user.

4.6.4. WCMComposer

This class is used to get content inside the WCM product. You should not access content 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 workspace, String nodeIdentifier, HashMap<String, String> filters, SessionProvider sessionProvider)throws Exception;	`workspace`
	`nodeIdentifier`
	`filters`
	`sessionProvider`: The session provider.
		`Node`	Return content at the specified path based on filters.

getContents(String workspace, String path, HashMap<String, String> filters, SessionProvider sessionProvider)throws Exception;	`workspace`
	`path`
	`filters`
	`sessionProvider`: The session provider.
		`List<Node>`	Return content at the specified path based on filters.

updateContent(String workspace, String nodeIdentifier, HashMap<String, String> filters)throws Exception;	`workspace`
	`path`
	`filters`
		`boolean`	Update content.

updateContents(String workspace, String nodeIdentifier, HashMap<String, String> filters)throws Exception;	`workspace`
	`path`
	`filters`
		`boolean`	Update content.

getPaginatedContents(Node nodeLocation,HashMap<String, String> filters, SessionProvider	`nodeLocation`: The content location. `filters`	`boolean`	Return content at the specified path based on filters.

Method	Param	Return	Description
sessionProvider) throws Exception;	`sessionProvider`: The session provider.		
getAllowedStates(String mode)throws Exception;	`mode`	`List<String>`	Return the 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;	`N/A`	`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.

4.6.5. NewFolksonomy

This service is used to manage all tags and their styles. Currently, it just supports adding/editing/removing the Private & Public tags.

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 folksonomy link to the node in the tag. `repository`: The repository name. `workspace`: The workspace name. `userName`: The username.	`void`	Add a private tag to a document. A folksonomy link will be created in a tag node.
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.	`void`	Add a group tag to a document. A folksonomy link will be created in a tag node.

Method	Return	Prototype	Description
	`repository`: The repository name. `workspace`: The workspace name. `roles`: The user roles.		
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.

`workspace`: The workspace name.| `void` | Add a public tag to a document. A folksonomy link will be created in a tag node.

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.	`List<Node>`	Get all private tags.

	`workspace`: The workspace name.		
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.
getAllGroupTagsString[] roles, String repository, String workspace) throws Exception ;	`roles`: The roles of user. `repository`: The repository name. `workspace`: The workspace name.	`List<Node>`	Get all tags by groups.
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.
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(Strir tagPath, Node document, String repository, String workspace) throws Exception;	`treeName`: The name of a folksonomy tree. `tagName`: The name of a tag.	`void`	Remove a tag of a given document.

	`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(documentNode, String repository, String workspace) throws Exception;	`documentNode`: The document node.
	`repository`
	`workspace`
		`List<Node>`	Get all tags linked to a given document.

getLinkedTagsOfDocumentBscope, String value, Node documentNode, String repository, String workspace) throws Exception;	`documentNode`: The document node. `repository`: The repository name. `workspace`: The workspace name.	`List<Node>`	Get all tags linked to a given document by scope.

removeTagsOfNodeRecursivnode, String repository, String workspace, String username, String groups) throws Exception;	`node`Node
	`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.

4.6.6. 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(PortletTemplatePlu portletTemplatePlugin) throws Exception	`portletTemplatePlugin`	`void`	Add the plugin..
getAllManagedPortletName(S repository) throws Exception	`repository`	`List<String>`	Retrieve all the portlet names that have dynamic groovy templates managed by service.

getTemplatesByApplication(repository, String portletName, SessionProvider provider)throws Exception;	`repository`
	`portletName`
	`provider`: The provider.
		`List<String>`	Retrieve the templates node by application.

getTemplatesByCategory(St 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	`repository` `portletName`	`node`	Retrieve the template by name.

Method	Param	Return	Description

templateName, SessionProvider sessionProvider)throws Exception;	`category`
	`templateName`
	`sessionProvider`		

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 templateName, SessionProvider sessionProvider)throws Exception;	`repository`
	`portletName`
	`catgory`
	`templateName`
	`sessionProvider`
		`void`	Remove the template.

4.6.7. 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.	`node`	Return the node at relPath related to the ancestor node. If the node is a link and giveTarget has been set to <code>true</code>, the target node will be returned.

Method	Param	Return	Description
	`giveTarget`: Indicate if the target must be returned in case the item is a link.		
getItem(String repository, String workspace, String absPath) throws PathNotFoundException, RepositoryException;	`repository`: The repository name. `workspace`: The workspace name. `absPath`: An absolute path.	`item`	Return the item at the specified absolute 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	`session`: The session is used to get the item.	`item`	Return the item at the specified absolute path.

Method	Param	Return	Description
PathNotFoundException, 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.	`boolean`	Return <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.

4.6.8. 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. Deprecate: This method is not supported by JODConverter 3.0 anymore, please use *convert(File, File, String)* instead.

	`input`	`boolean`	Convert input File to output File with the outputFormat.

Method	Param	Return	Description
convert(File input, File output, String outputFormat) throws OfficeException;	`output` `outputFormat`		

4.6.9. TimelineService

TimelineService is used to get all documents by time frame (day, month, year).

Method	Param	Return	Description

getDocumentsOfToday(Strin nodePath,String repository,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`repository`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents of today. (Deprecated)

getDocumentsOfToday(Strin nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents of today.

getDocumentsOfToday(Strin nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser, boolean isLimit) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
	`isLimit`
		`List<Node>`	Get all documents of today.

getDocumentsOfYesterday(S nodePath,String repository,String workspace,	`nodePath`	`List<Node>`	Get all documents of yesterday. (Deprecated)

Method	Param	Return	Description

SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`repository`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`		

getDocumentsOfYesterday(S nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents of yesterday.

getDocumentsOfYesterday(S nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser, boolean isLimit) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
	`isLimit`
		`List<Node>`	Get all documents of yesterday.

getDocumentsOfEarlierThisW nodePath,String repository,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePathg`
	`repository`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents from earlier this week to yesterday. (Deprecated)

getDocumentsOfEarlierThisW nodePath,String workspace, SessionProvider	`nodePathg` `workspace`	`List<Node>`	Get all documents from earlier this week to yesterday.

Method	Param	Return	Description

sessionProvider, String userName, boolean byUser) throws Exception;	`sessionProvider`
	`userName`
	`byUser`		

getDocumentsOfEarlierThisW nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser, boolean isLimit) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
	`isLimit`
		`List<Node>`	Get all documents from earlier this week to yesterday.

getDocumentsOfEarlierThisM nodePath,String repository,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`repository`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents from earlier this month to earlier this week. (Deprecated)

getDocumentsOfEarlierThisM nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents from earlier this month to earlier this week.

getDocumentsOfEarlierThisM nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser, boolean isLimit) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
		`List<Node>`	Get all documents since earlier this month to earlier this week.

Method	Param	Return	Description

	`userName`
	`byUser`
	`isLimit`		

getDocumentsOfEarlierThis(String nodePath,String repository,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`repository`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents from earlier this year to earlier this month. (Deprecated)

getDocumentsOfEarlierThis(String nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
		`List<Node>`	Get all documents from earlier this year to earlier this month.

getDocumentsOfEarlierThis(String nodePath,String workspace, SessionProvider sessionProvider, String userName, boolean byUser, boolean isLimit) throws Exception;	`nodePath`
	`workspace`
	`sessionProvider`
	`userName`
	`byUser`
	`isLimit`
		`List<Node>`	Get all documents from earlier this year to earlier this month.

getItemPerTimeline()	`N/A`	`int`	Get the number of documents per category displayed in the Timeline view.

4.6.10. SiteSearchService

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

Method	Param	Return	Description
addExcludeIncludeDataType(ExcludeIncludeDataTypePlugin plugin)	`plugin`: The plugin.	`void`	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<ResultNode>`	Find all child nodes whose contents match with the given keyword. These nodes will be put in the list of search results.

4.6.11. 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	`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	`metaModel`: The metadata of a page/content stored. `portalName`: The name of portal.	`void`	Remove the metadata of a page.

Method	Param	Return	Description
	`onContent`: Indicate whether the current page is the content page or the portal 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) throws Exception	`portalName`: The portal name.	`string`	Return Robots' content of 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.

4.6.12. ManageViewService

ManageViewService is used to work with views. This service has many functions which allow you to add, edit, delete, and get views.

Method	Param	Return	Description

addView(String name, String permissions, String template, List<?> tabs, String repository) throws Exception;	`name`
	`permissions`
	`template`
	`tabs`
	`repository`
		`void`	Insert a new view to the system. (Deprecated)

addView(String name, String permissions, String template, List<?> tabs) throws Exception;	`name`
	`permissions`
	`template`
		`void`	Insert a new view to the system.

Method	Param	Return	Description
	`tabs`		

getViewByName(String viewName, String repository, SessionProvider provider) throws Exception;	`viewName`
	`repository`
	`provider`
		`node`	Specify a new view depending on the view name. (Deprecated)

getViewByName(String viewName, SessionProvider provider) throws Exception;	`viewName` `provider`	`node`	Specify a new view depending on the view name.
getButtons() throws Exception;	`N/A`	`List<?>`	Return all strings of buttons.
removeView(String viewName, String repository) throws Exception;	`viewName` `repository`	`void`	Remove a view from the views list in the system. (Deprecated)
removeView(String viewName) throws Exception;	`viewName`	`void`	Remove a view from the views list in the system.
getAllViews(String repository) throws Exception;	`repository`	`List<ViewConfig>`	Return all views of the repository configured in the XML file. (Deprecated)
getAllViews() throws Exception;	`N/A`	`List<ViewConfig>`	Return all views of the repository configured in the XML file.
hasView(String name, String repository) throws Exception;	`name` `repository`	`boolean`	Return true if the given repository has a view. (Deprecated)
hasView(String name) throws Exception;	`N/A`	`boolean`	Return true if the given repository has a view.

getTemplateHome(String homeAlias, String repository, SessionProvider provider) throws Exception;	`homeAlias`
	`repository`
	`provider`
		`Node`	Get a template node that has the path. (Deprecated)

getTemplateHome(String homeAlias, SessionProvider provider) throws Exception;	`homeAlias` `provider`	`Node`	Get a template node that has the path.

getAllTemplates(String homeAlias, String repository, SessionProvider provider) throws Exception;	`homeAlias`
	`repository`
	`provider`
		`List<Node>`	Get all template nodes that have the path. (Deprecated)

Method	Param	Return	Description
getAllTemplates(String homeAlias, SessionProvider provider) throws Exception;	`homeAlias` `provider`	`List<Node>`	Get all template nodes that have the path.

getTemplate(String path, String repository, SessionProvider provider) throws Exception;	`path`
	`repository`
	`provider`
		`Node`	Return a node that has the path of the repository. (Deprecated)

getTemplate(String path, SessionProvider provider) throws Exception;	`path` `provider`	`Node`	Return a node that has the path of the repository.

addTemplate(String name, String content, String homePath, String repository) throws Exception;	`name`
	`content`
	`homePath`
	`repository`
		`String`	Insert a new template to a node by specifying its path. (Deprecated)

addTemplate(String name, String content, String homePath) throws Exception;	`name`
	`content`
	`homePath`
		`String`	Insert a new template to a node by a specified path.

updateTemplate(String name, String content, String homePath, String repository) throws Exception;	`name`
	`content`
	`homePath`
	`repository`
		`String`	Update a template for a node by specifying its path. (Deprecated)

updateTemplate(String name, String content, String homePath) throws Exception;	`name`
	`content`
	`homePath`
		`String`	Update a template for a node by specifying its path.

removeTemplate(String templatePath, String repository) throws Exception;	`templatePath` `repository`	`void`	Remove the template from the given node by specifying its path. (Deprecated)
removeTemplate(String templatePath) throws Exception;	`templatePath`	`void`	Remove the template from the given node by specifying its path.

Method	Param	Return	Description

addTab(Node view, String name, String buttons) throws Exception ;	`view`
	`name`
	`buttons`
		`void`	Insert a new tab to the given view node.

init(String repository) throws Exception ;	`repository`	`void`	Get all templates that are configured in the XML file of a specified repository. (Deprecated)
init() throws Exception ;	`N/A`	`void`	Get all templates that are configured in the XML file of a specified repository.

4.7. Deprecated portlets

In Content, there are some deprecated portlets, including **Browse Content** (BC), **Parameterized Content Viewer** (PCV), **Parameterized Content List Viewer** (PCLV), **Category Navigation** (CN), **Newsletter Viewer**, **Newsletter Manager**, **Form Builder**.

In which:

- The **BC**, **Newsletter Viewer**, **Newsletter Manager** and **Form Builder** portlets are not used anymore.
- The **PCV** portlet is still used, but its java class named *UIPCVPortlet* is replaced by *UISingleContentViewerPortlet* of the **Single Content Viewer** (SCV) portlet.
- The **PCLV** portlet is still used, but its java class named *UIPCLVPortlet* is replaced by *UICLVPortlet* of the **Content List Viewer** (CLV) porlet.
- The **CN** portlet is still used, but its java class named *UICategoryNavigationPortlet* is replaced by *UICLVPortlet* of the **CLV** porlet.

See also

- WCM Templates
- WCM Explorer
- Extensions
- CMIS Usage code examples
- Public REST APIs
- Public Java APIs
- Miscellaneous and Tips
- FAQs

4.8. Miscellaneous and Tips

xCMIS project links:

- Community site
- JIRA site
- Forum

- xCMIS latest news
- eXo Platform blog
- Demo site with CMISExpert client

CMIS-related links:

- CMIS specification
- CMIS working group (list, mail, feeds, and more)
- CMIS clients

See also

- WCM Templates
- WCM Explorer
- Extensions
- CMIS Usage code examples
- Public REST APIs
- Public Java APIs
- Deprecated portlets
- FAQs

4.9. FAQs

This section provides FAQs related to the product.

Q1. 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.

- **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>
```

- **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.

See also

- WCM Templates
- WCM Explorer
- Extensions
- CMIS Usage code examples
- Public REST APIs
- Public Java APIs
- Deprecated portlets
- Miscellaneous and Tips