DITACMS

What's New in DITA CMS 4.3

Last revised: June 13, 2017

Table of contents

Chapter 3: New Scheduler 4.3 features

Copyright notice

1 New DITA CMS 4.3 features

Topics:

This section describes the new features introduced in this release of the DITA CMS.

End of Support

The IXIASOFT Web Author and Web Collaborative Review modules are being merged into a unique DITA CMS Web component to take your content creation and reviewing processes to the next level. The DITA CMS Web component will include the authoring and reviewing features available today as well as new features, such as a map view, topic and map creation, as well as map editing.

As such, all support for Web Author and Web Collaborative Reviewer will stop by December 31st, 2017.

Support for DITA 1.3

DITA CMS now supports version 1.3 of the DITA specification.

Some of the new DITA 1.3 features include:

- Scoped keys
- Troubleshooting topic type
- Enhancements to conditional processing
- Increased support for producing user assistance
- New domains

To upgrade your DITA CMS 4.3 installation to version 1.3 of the DITA specification, see the *Upgrading a Deployment to DITA 1.3* document.

Reporting issues to IXIASOFT

If you encounter issues with the DITA CMS, you can create an error package that you send to IXIASOFT for troubleshooting.

This error package contains logs and data that will help IXIASOFT troubleshoot the issue that you encountered. For example, the error package includes:

- A screenshot of the DITA CMS
- The DITA CMS system configuration, including the index definition document
- The *UItrace.log* file, if available
- The *DITA_CMS-<date>.timers* file, if available
- The *eclipse.ini* file
- The *DITA_CMS.log* debug log, if the application is running in debug mode
- The Eclipse log file

This information is saved in the following file:

DITACMS_ERROR_REPORT_<user_name>_[<email_address]_<date>_<time>.zip

For example:

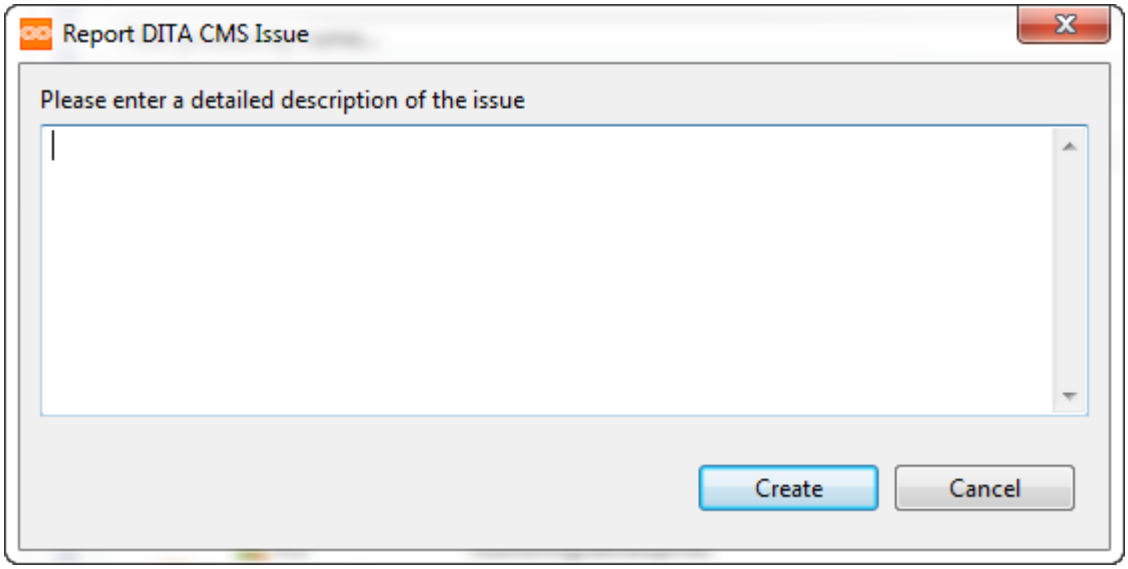
DITACMS_ERROR_REPORT_John_Smith_[john.smith@acme.com]_20151125_160009.zip

You can then attach this error package to your OTRS ticket.

To create the error package:

1. **From the DITA CMS toolbar, select *Help > Report DITA CMS Issue*.**

The Report DITA CMS Issue window is displayed:



2. **Enter a description of the issue you encountered.**

   Add as much information as possible, including:

   - What you were trying to do
   - The view or perspective you were working with
   - The expected behavior
   - The error that occurred

3. **Click Create.**

   The error package is prepared and the **Save as** window appears.

4. **Browse to the location where you want to save the file and click Save.**
5. **Open an OTRS ticket and attach the error package to the ticket.**

Configure the incremental localization process

The incremental localization process allows you to start the localization process without having all the objects in the Authoring:done status (or the equivalent status in your deployment).

> **Important:** The incremental localization process is applicable only to the sequential localization model.
>
> **Note:** If you localize individual topics and images, IXIASOFT strongly recommends that you implement the incremental localization process.

Normally, the map and all its children should be in the Authoring:done status (or the equivalent in your deployment) before you can start the localization process. When you implement the

incremental localization process, you define what other statuses are acceptable before the objects can be sent to the Localization cycle. For example, instead of having the map, the topics, and all the images in Authoring:done before starting the localization process, you can configure it so you can begin the localization process even if some of the topics are still in Authoring:work.

To incorporate the incremental localization process in the workflow, you need make the following changes in the configuration files:

- A new status called `Localization:do not translate` must be added to the configuration files which define the statuses available for each object. The Localization:do not translate status identifies the objects that are not authorized to be translated.
- In the `Localize` action in the access rights configuration file, you must define the statuses objects must be set to before they can be sent to the Localization cycle.
- In the `localize_api` action in the access rights configuration file, you must define which statuses for each object are set to Localization:tb translated when they are sent to the Localization cycle. All statuses not defined in the localize_api action will automatically be set to Localization:do not translate when they are sent to the Localization cycle.
- In the `Prepare kit` action in the access rights configuration file, you define which objects in which statuses are included in the localization kit.

If objects in the Localization:do not translate status are included in the localization kit when it is prepared, DITA CMS automatically adds the `translate="no"` attribute to the root element of those objects so translation vendors can identify which files are not to be localized, and removes all the `ixia_locid` attributes from the elements contained in the objects to ensure that they cannot be imported after the localized files are received.

To configure the incremental localization process:

1. **Open the TEXTML Administration perspective by clicking the TEXTML Administration shortcut on the tool bar. If the shortcut is not displayed, follow these steps:**
   a) **Select *Window > Open Perspective > Other***
   b) **Click TEXTML Administration.**
   c) **Click OK.**
2. **In the TEXTML Administration view, double-click the server. If your server is not displayed in the view, you must add it to the view.**
3. **When the Connect as dialog opens, type your username and password and click OK.**
4. **Double-click the name of your docbase to open a connection to the Content Store.**
5. **Locate the *status.xml* file in the repository's */system/conf* collection.**
6. **Right-click the file and click Check Out.**

7. **Double-click the file to open it in the editor.**
8. **In the section `<collection="/content/localization/" >`, add the `do not translate` state (do not alter the name) by copying the following lines and pasting them inside the `states` element.**

```
<state level="0" name="do not translate" type="work">
 <lockable>
  <objtypes>
   <type>none</type>
  </objtypes>
 </lockable>
 <milestone/>
 <nextStates/>
</state>
```

Therefore, it should look like the following example:

```
<cycle collection="/content/localization/" description="" initial="false"
lastworkcycle="false"
 level="0" name="Localization" type="localization">
 <nextCycle/>
 <states>
  <state level="0" name="do not translate" type="work">
   <lockable>
    <objtypes>
     <type>none</type>
    </objtypes>
   </lockable>
   <milestone/>
   <nextStates/>
  </state>
  <state initial="true" level="0" name="tb translated" type="work">
   <lockable>
    <objtypes>
     <type>image</type>
    </objtypes>
   </lockable>
   <nextStates>
    <next>in translation</next>
   </nextStates>
  </state>

  [ . . . ]

 </states>
</cycle>
```

9. **Repeat the previous steps you completed for *status.xml* to add the new state to the following files:**
   - map_status.xml
   - topic_status.xml
   - image_status.xml
10. **Save, close, and check in the files.**
11. **Open the Access Manager window.**

a) **Right-click the Content Store.**
b) **Click DITA CMS.**
c) **Click Manage Access.**

12. **In the Actions column, click Localize and click Lock to configure the localization process for DITA CMS.**
13. **For each condition, define which objects in which statuses can be sent to the Localization cycle:**
    a) **In the Conditions column, click a condition.**
    b) **Click the arrow next to an object to expand the list of cycles.**
    c) **Click the arrow next to Authoring to expand the list of statuses.**
    d) **Select each status which is authorized to be sent to the Localization cycle.**
    e) **Repeat for each object.**
    f) **Repeat for each condition.**

    For example, if you wanted to allow the map and all its children to be able to be sent to the Localization cycle regardless their status, you would select all the statuses for all the objects.

14. **In the Actions column, click localize_api to define which statuses are set to Localization:tb translated (or the equivalent status in your deployment) when objects are sent to the Localization cycle. All other statuses will be set automatically to Localization:do not translate.**
15. **For each condition, define which statuses can be sent through the localization process:**
    a) **In the Conditions column, click a condition.**
    b) **Click the arrow next to an object to expand the list of cycles.**
    c) **Click the arrow next to Authoring to expand the list of statuses.**
    d) **Select each status for which you want to authorize to be translated.**
    e) **Repeat for each condition.**

    For example, if you only wanted objects in Authoring:done and Authoring:accepted (or the equivalent in your deployment) to be translated, you would select only those statuses. This means that objects in the Authoring:done and Authoring:accepted statuses will be set to Localization:tb translated when they are sent to the Localization cycle. All objects in any other status will be set to Localization:do not translate.

16. **In the Actions column, click Prepare kit.**
17. **For each condition, define which objects in which statuses are included in the localization kit:**
    a) **In the Conditions column, click a condition.**

b) **Click the arrow next to an object to expand the list of cycles.**
c) **Click the arrow next to Localization to expand the list of statuses.**
d) **Select each status you want included in the localization kit.**
e) **Repeat for each object.**
f) **Repeat for each condition.**

For example, if you only want to include the objects that you want translated in the kit, you would select only Localization:tb translated (or the equivalent in your deployment).

18. **In the Actions column, click Retranslate.**
19. **For each condition, define which objects can be retranslated:**
    a) **In the Conditions column, click a condition.**
    b) **Click the arrow next to an object to expand the list of cycles.**
    c) **Click the arrow next to Localization to expand the list of statuses.**
    d) **Select each status you want included in the localization kit.**
    e) **Clear the checkbox beside do not translate.**
    f) **Repeat for each condition.**

    Tip: It is recommended to selected all the statuses, except **do not translate**.
20. **When you are done, click CheckIn Document to commit the changes to the access rights back to the Content Store.**
21. **Test your implementation. If it is not behaving as expected, verify the access rights are configured correctly for each object and status in each condition.**
22. **Inform users of the changes.**

    The changes will be applied automatically once users close and then reopen their DITA CMS. Users can also apply the changes without restarting their DITA CMS by clicking ***DITA CMS*** > ***Synchronize Configuration***.

Example of implementing an incremental localization process

The example is intended to illustrate an implementation of the incremental localization process. Your configuration and workflow may differ significantly.

For this example, you want to implement an incremental localization process that allows you to send a map and all its children through localization even though some of the objects are not finished. Suppose your current Authoring cycle contains these statuses for all objects: Authoring:work, Authoring:review, Authoring:complete, and Authoring:approved; and your Localization cycle contains Localization:tb translated, Localization:in translation,

Localization:review, and Localization:done. You decide that you want objects in the Authoring:approved and Authoring:done to be translated, but not objects in Authoring:work or Authoring:review. This would allow you to start localizing some of the document as it becomes ready while continuing to work on other parts. Also, when you prepare the kit, you decide that you want to include all the objects (map and its children) regardless of their status in the localization kit to provide the translators with context for the content being localized.

If you were implementing this example, you would:

- Add the new Localization:do not translate state to the Localization cycle.
- Modify the access rights to allow any object in any state in the Authoring cycle to be sent to the Localization cycle.
- Modify the access rights to define that only objects in Authoring:complete and Authoring:approved can be set to Localization:tb translated when they are sent to the Localization cycle.
- Modify the access rights to allow all objects regardless of their status to be included in the localization kit.

To add the new state to the Localization cycle, you would lock the *status.xml*, *map_status.xml*, *topic_status.xml*, and *images_status.xml* configuration files and add the `do not translate` state to the `<collection="/content/localization/" >` section in each file.

After adding the new state, you would right-click the Content Store, select ***DITA CMS*** > ***Manage Access*** to open the **Access Manager** window. In the **Access Manager** window, you will edit the access rights configure the incremental localization process.

In the `Localize` action, you define the statuses objects must be set to before they can be sent to the Localization cycle. Since you want to allow any object in any state in the Authoring cycle to be sent to the Localization cycle in this example, then for each object in the status/type column you would need to expand `Authoring` and select the checkboxes beside each state.

In the `localize_api` action, you define which statuses for each object are set to Localization:tb translated when they are sent to the Localization cycle. Since you only want Authoring:complete and Authoring:approved to be translated in this example, you would expand `Authoring` and select the checkboxes beside `complete` and `approved`.

In the `Prepare kit` action, you define which objects in which statuses are included in the localization kit. Since you want to include everything in the kit for this example, you would expand `Localization` under each object and select the checkboxes beside all the states.

With it configured like this, the process would work like this. When you right-click the map and click ***Localization*** > ***Localize***, the map and all its children are sent to the Localization cycle

regardless of their status. In the Localization cycle, objects that were in Authoring:complete and Authoring:approved are set to Localization:tb translated and objects that were in Authoring:work and Authoring:review are set to Localization:do not translate.

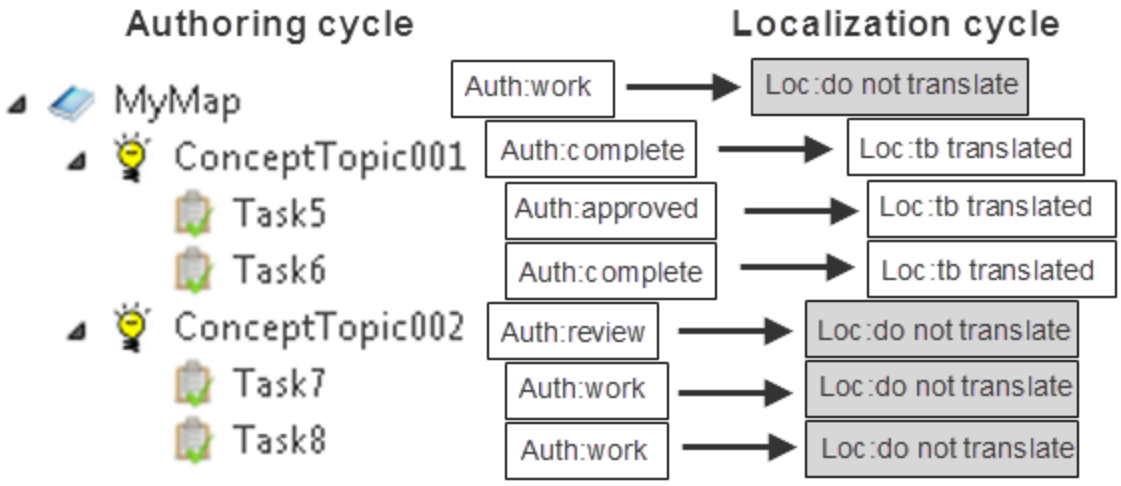


Figure 1: Example of sending a map to the Localization cycle with incremental localization

When you prepare the localization kit, all the objects (map and its children) regardless of their status are included in the kit. The objects in Localization:tb translated are set to Localization:in translation. For each object in the Localization:do not translate status, DITA CMS automatically adds the `translate="no"` attribute to the root element and removes all the `ixia_locid` attributes from the elements it contains. When you perform the process to import the files after they return from translation, the objects in the Localization:in translation status are set to the Localization:review status and the objects in Localization:do not translate are not imported.

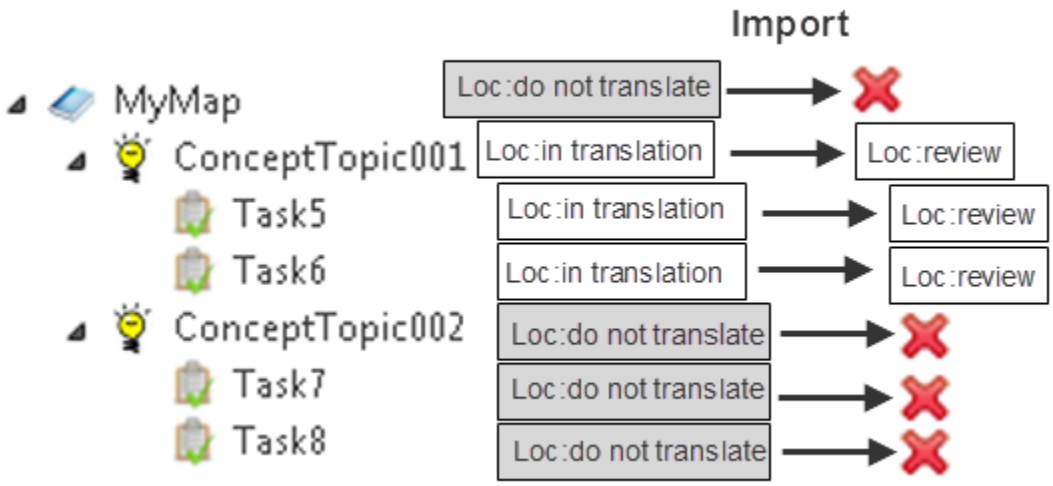


Figure 2: Example of importing localized objects

Localizing from an alternate language (pivot language localization)

A pivot language is used as intermediate language when a direct translation between the source language and the target language is not available.

This may occur when your translation vendor does not offer a direct translation from your source language or it is too costly to perform a direct translation. For example, you want to localize into Arabic from Japanese, but your translation vendor does not offer Japanese to Arabic (or it is too expensive); however, they do offer Japanese to English and English to Arabic. This means that you need to use English as a pivot language so you can create a sequence to localize Japanese into English and then English into Arabic.

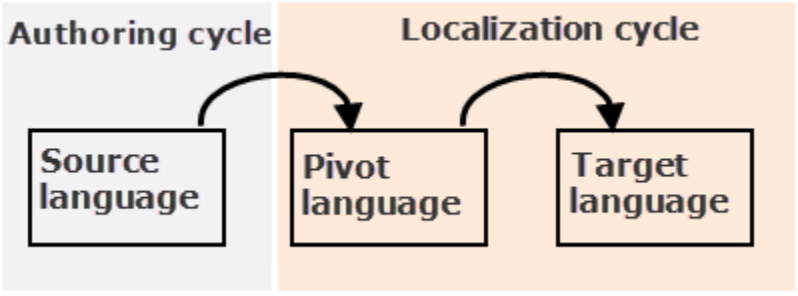


Figure 3: Pivot Language localization sequence

When pivot language localization is enabled, the DITA CMS administrator configures which languages can be localized into which languages. Although an object can be localized in any

available language as defined in the configuration files, once an object has been localized from a pivot language, you cannot shortcut its localization sequence. For example, if you have localized a map and its child objects from Japanese to English to Arabic, you **cannot** go through the cycle of updating **that map and its child objects** from Japanese directly to Arabic.

When a source object is translated for the first time into a language, DITA CMS creates a new object in the target language (referred to as a *language object*). This language object has its own **Revision** number as well as an **Authoring Revision** property, which stores the revision number of the source object when it was sent for translation. With pivot language localization, the source object is not necessarily in the Authoring cycle. For example, if you have localized from Japanese to English to Arabic to Indonesian, the revision number stored as the Authoring Revision for the Indonesian language object comes from its source object (the Arabic language object), not the originally authored content (the Japanese object).

Configuring pivot language localization

Pivot language localization is an optional feature configurable as part of the sequential localization method, which allows the localization of objects from the Localization cycle.

Localization typically starts from a single source language in the Authoring cycle directly producing one or more target languages in the Localization cycle. In some cases you may not be able to perform a direct translation from your source language to your target language and instead need to translate into an intermediate language before proceeding to your target language. This intermediate language is referred to as a pivot language. To localize from a pivot language, you must configure pivot language localization in DITA CMS. When you start using pivot language localization, you will have one or more source languages in the Localization cycle in addition to the one in the Authoring cycle with each producing one or more target languages.

You configure pivot localization by defining all the source and pivot languages and their target languages in the *pivotLanguages.xml* file. In this file the pivot language is treated like just another source language. As a result, the options available in the dialog box when you right-click an object and select ***Localization*** > ***Localize*** are limited to the target languages configured and available for that source language.

For example, if your authoring language is English and you localize from English to French, Japanese, and German and use French as a pivot language for Arabic, you would need to configure the following:

- English as a source language with its target languages of French, Japanese, and German
- French as a source language with its target language of Arabic

Following this example once it was configured, when you right-click an English map, the options displayed in the **Select Languages** dialog box would be limited to French, Japanese, and German instead of offering all the languages configured in the *languages.xml* file.

BEFORE YOU BEGIN:

Choose your pivot languages carefully. The configuration choices you make will have far-reaching and long-term effects on your content:

1. Consider the long-term translation cycle. Once an object has been localized from a pivot language, you cannot shortcut the sequence using a different localization sequence. For example, if you have localized a map and its child objects from Japanese to English to Arabic, you **cannot** go through the cycle of updating **that map and its child objects** from Japanese directly to Arabic.
2. Once a map has been localized using a particular localization sequence, it and its child objects **cannot** be translated back into any of the languages already used in the localization sequence. For example, if you have localized a map and its child objects from Japanese to English to Arabic, you cannot localize from Arabic back to English or Japanese. When you select ***Localization*** > ***Localize***, your choices will be limited to the available target languages configured in the *pivotLanguages.xml* file minus the languages already used in the localization sequence.
3. Consider where objects will be shared and reused. You cannot mix objects that were localized using different sequences in the same map. For example, if you have configured a sequence of Japanese to English to Arabic and also a sequence of French to Arabic, you cannot mix topics localized using the Japanese to English to Arabic sequence in a map containing topics localized using the French to Arabic sequence.

To configure pivot language localization:

1. **Open the TEXTML Administration perspective by clicking the TEXTML Administration shortcut on the tool bar. If the shortcut is not displayed, follow these steps:**
   a) **Select *Window > Open Perspective > Other***
   b) **Click TEXTML Administration.**
   c) **Click OK.**
2. **In the TEXTML Administration view, double-click the server. If your server is not displayed in the view, you must add it to the view.**
3. **When the Connect as dialog opens, type your username and password and click OK.**
4. **Double-click the name of your docbase to open a connection to the Content Store.**
5. **Configure your localization workflow.**

Restriction: Pivot language localization is only available with the sequential localization model.

6. **Open the Access Manager window.**
   a) **Right-click the Content Store.**
   b) **Click DITA CMS.**
   c) **Click Manage Access.**
7. **In the Actions column, click Localize and click Lock.**
8. **For each condition, define which objects in which statuses can be localized:**
   a) **In the Conditions column, click a condition.**
   b) **Click the arrow next to an object to expand the list of cycles.**
   c) **Click the arrow next to Localization to expand the list of statuses.**
   d) **Select each status which is authorized to be localized.**
   e) **Repeat for each object.**
   f) **Repeat for each condition.**

   For example, if you only wanted objects in Localization:done (or the equivalent in your deployment) in the Localization cycle to be a pivot language, you would select only that status. This means that objects in the Localization:done status will have the ***Localization*** > ***Localize*** right-click menu enabled depending on your configuration.

9. **In the Actions column, click localize_api to define which statuses are set to Localization:tb translated (or the equivalent status in your deployment) when you click the Localize command.**
10. **For each condition, define which statuses in the Localization cycle can be sent through the localization process:**
    a) **In the Conditions column, click a condition.**
    b) **Click the arrow to expand the list of cycles.**
    c) **Click the arrow next to Localization to expand the list of statuses.**
    d) **Select each status for which you want to authorize to be translated.**
    e) **Repeat for each condition.**

    For example, if you only wanted objects in Localization:done (or the equivalent in your deployment) to be translated, you would select only that status. This means that objects in the Localization:done status will be set to Localization:tb translated when they are sent to be localized.

11. **Click CheckIn Document to commit the changes to the access rights back to the Content Store.**
12. **Locate the** *pivotLanguages.xml* **file in the repository's** */system/conf* **collection.**
13. **Right-click the file and click Check Out.**
14. **Double-click the file to open it in the editor.**
15. **Define the source and target languages using the template in the** *pivotLanguages.xml* **file. Use the language codes defined in the** *languages.xml* **file.**

    For example, if you were translating from English to French, Japanese, German and using French as a pivot language for Arabic, you would configure the following in the *pivotLanguages.xml* file:

```
<pivotlanguages>
 <sourcelanguage name="en-us">
  <targetlanguage>fr-fr</targetlanguage>
  <targetlanguage>ja-jp</targetlanguage>
  <targetlanguage>de-de</targetlanguage>
 </sourcelanguage>
 <sourcelanguage name="fr-fr">
  <targetlanguage>ar-ma</targetlanguage>
 </sourcelanguage>
</pivotlanguages>
```

16. **When you are done, click CheckIn Document to commit the changes to the access rights back to the Content Store.**
17. **Test your implementation. If it is not behaving as expected, verify the access rights are configured correctly for each object and status in each condition.**
18. **Inform users of the changes.**

    The changes will be applied automatically once users close and then reopen their DITA CMS. Users can also apply the changes without restarting their DITA CMS by clicking ***DITA CMS*** > ***Synchronize Configuration***.

Localizing from a pivot language

Pivot language localization starts by sending an already localized map and its contents to be localized into another language.

A pivot language is used as intermediate language when a direct translation between the source language and the target language is not available to you for whatever reason. When pivot language localization is configured, the source languages and their allowed target languages are explicitly defined in the configuration files. As a result, the options available in the dialog box when you right-click an object and select ***Localization*** > ***Localize*** are limited to the target languages configured for that source language.

Important: The localization process requires a substantial amount of system resources. If you are localizing several maps in many different languages you may notice an impact on your computer's performance.

To start the localization process from a pivot language:

1. **Search for the map or topic you want to use as the source in the Localization cycle.**

   Remember: The source map or topic must be in a valid source language and at a valid status for localization.

2. **In the Search Results view, right-click the required map or topic and select *Localization > Localize...* from the menu.**

   The **Select Languages** dialog appears offering only the valid target languages. Your choices will be limited to the target languages configured for that language minus the languages already used in the localization sequence.

3. **In the Select Languages dialog, select the required target language(s).**
4. **Click Create.**

   The **Localize** dialog appears as copies of the map and its topics are being created in the Localization cycle, one for each of the target languages.

The map and its topics now have the status Localization:tb translated (or its equivalent in your workflow).

Next steps in the localization process:

- Create an image localization kit for the graphics, if required.
- Create a localization kit for the map and its topics.

Create a subject scheme map

You can create a subject scheme map to define a list of controlled values for the conditional processing attributes (enumerated attribute values subject scheme map).

A subject scheme allows you to create a custom set of conditional processing attribute values without having to create a DITA specialization for the attribute. In the subject scheme map you define a controlled list of values and bind them to an existing conditional processing attribute. The subject scheme map can then be added to a ditamap, which then controls what conditional processing attribute values are available and displayed for that ditamap and its child objects while the ditamap is displayed in the DITA Map view.

To create a subjectScheme map:

1. **Create a new subjectScheme by following the procedure for creating a map. In the Map template box, select the subjectScheme template that you want to use.**
2. **Open the subjectScheme map in the XML editor.**

   The template contains some example values you can use as a guide to create your own. For example:

```
<!-- Define user values  -->
 <subjectdef keys="user">
  <subjectdef keys="Marketing"/>
  <subjectdef keys="TechPubs"/>
 </subjectdef>
<!-- Bind @audience to user values  -->
 <enumerationdef>
  <attributedef name="audience"/>
  <subjectdef keyref="user"/>
 </enumerationdef>
```

3. **Define your list of controlled values.**
   a) **Locate the `subjectdef` element. In the `keys` attribute, type a name for your list of controlled values.**

      For example, you would change the "user" value from the template to one of your choosing:

```
<subjectdef keys="user">
 <subjectdef keys="Marketing"/>
 <subjectdef keys="TechPubs"/>
</subjectdef>
```

   b) **Define each value you want using the `subjectdef` element nested inside the parent `subjectdef` element.**

      For example, you would change the "Marketing" and "TechPubs" values from the template to ones of your choosing as well as add as many new values as you need:

```
<subjectdef keys="user">
 <subjectdef keys="Marketing"/>
 <subjectdef keys="TechPubs"/>
</subjectdef>
```

4. **Bind the list of controlled values to a conditional processing attribute.**
   a) **Locate the `enumerationdef` element.**

      For example:

```
<enumerationdef>
 <attributedef name="audience"/>
 <subjectdef keyref="users"/>
</enumerationdef>
```

b) **Define the conditional processing attribute you want to bind using the `attributedef` element. Replace the value for the `name` attribute with the name of the conditional processing attribute that you want to use.**

For example, you would change the "audience" value from the template with any of the conditional processing attributes.

```
<enumerationdef>
 <attributedef name="audience"/>
 <subjectdef keyref="users"/>
</enumerationdef>
```

c) **Define the name of the list to be bound to the conditional processing attribute using the `subjectdef` element. Replace the value in the `keyref` attribute with the name you defined for your controlled list of values.**

For example, you would change the "users" value from the template to the one you defined.

```
<enumerationdef>
 <attributedef name="audience"/>
 <subjectdef keyref="users"/>
</enumerationdef>
```

5. **Save and release the subjectScheme map.**
6. **Add the subjectScheme map to other ditamaps as required.**

Miscellaneous improvements to DITA CMS

DITA CMS 4.3 includes the following miscellaneous improvements:

- **Redline With enhancements** - You can now compare a map with a snapshot or published map in the Authoring or Localization cycle.
- **Use of wildcards in perspective names** - When specifying perspective names in the *system/conf/eclipseui.xml file* to simplify the interface, you can now use wildcards to specify multiple perspectives in a single line. For example, "com.ixiasoft.dita.eclipse.gui.perspective.*" will match all the DITA CMS perspectives.

2 New DRM 4.3 features

Topics:

This section describes the new DRM feature introduced in this release of the DITA CMS.

Using multi-level libraries

Multi-level libraries allow documentation organizations to organize their content libraries in layers so that they match the product architecture.

This section describes how to use multi-level libraries in your deployment.

Configure the layers for your deployment

You configure the layers for your multi-level libraries by editing the *library-layers.xml* configuration file.

In the initial configuration, a single layer is available ("`library default layer`" with the id "`1`") , as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library PUBLIC "-//ixiasoft.com//cms//config//libraryLayers" "librarylayers.dtd">
<!--
=========================================================================================================
 -->
<!-- -->
<!-- FILE: librarylayers.xml -->
<!-- -->
<!-- DESCRIPTION: This file defines the available library layers within the docbase. -->
<!-- Note that layer ID 0 is the Product layer and cannot be modified. Library layer IDs
start at 1.-->
<!-- -->
<!--
=========================================================================================================
 -->
<!-- REVISION HISTORY: -->
<!-- -->
<!--
=========================================================================================================
 -->
<library>
    <layer id="1" name="library" description="library default layer" isDefault="yes"/>
</library>
```

To configure layers, you create one `<layer>` element per layer. A layer has the following attributes:

Table 1: Attributes of the layer element

Attribute	Description
`id`	Level of the library layer. Library layers start at 1, the highest level. You can add as many layers as necessary. A library or product can reference content from a library that is at the same level (sibling library) or lower (child library).

Attribute	Description
	The level must be unique within the *library-layers.xml* configuration file.
`name`	Name of the library layer. This name is used in the DRM dialogs to identify the layer. Make sure to enter a descriptive name.
`description`	Description of the library layer.
`isDefault`	Specifies whether this layer is the default layer. Only one layer can be set as the default (`isDefault="yes"`). The default layer is used for backward compatibility. When an existing library version is released, if it does not have a layer level associated to it, it will be assigned this default layer.

You can add other layers, as appropriate for your deployment. Remember that creating multiple layers will increase the complexity of the DRM.

> **Note:** Please contact your IXIASOFT DITA specialist before implementing multi-level libraries in your deployment, as they increase the complexity of the DITA CMS. Multi-level libraries must be planned carefully.

To configure the layers for your multi-level libraries:

1. **Open the TEXTML Administration perspective by clicking the TEXTML Administration shortcut on the tool bar. If the shortcut is not displayed, follow these steps:**
   a) **Select *Window > Open Perspective > Other***
   b) **Click TEXTML Administration.**
   c) **Click OK.**
2. **In the TEXTML Administration view, double-click the server. If your server is not displayed in the view, you must add it to the view.**
3. **When the Connect as dialog opens, type your username and password and click OK.**
4. **Double-click the name of your docbase to open a connection to the Content Store.**
5. **Locate the library-layers.xml file in the repository's *system/conf* collection.**
6. **Right-click the file and select Check Out.**
7. **Double-click library-layers.xml to open it in a text editor.**

8. **For each layer to add, add the following `<layer>` element:**

```
 <layer id="<layer_level>" name="<layer_name>" description="<layer_description>"
isDefault="yes|no"/>
```

For example:

```
 <layer id="2" name="Layer 2" description="Second library layer" isDefault="yes"/>
```

The following example shows the configuration for the library layers introduced in **Using multi-level libraries** on page 26:

```
<library>
    <layer id="3" name="Product Library Level" description="Product Library Level"
isDefault="yes"/>
    <layer id="6" name="Third Party Library" description="Third Party Library"
isDefault="no"/>
    <layer id="9" name="Operating System Library" description="Operating System Library"
isDefault="no"/>
</library>
```

9. **Save, close, and check in the library-layers.xml file.**
10. **Inform users of the changes.**

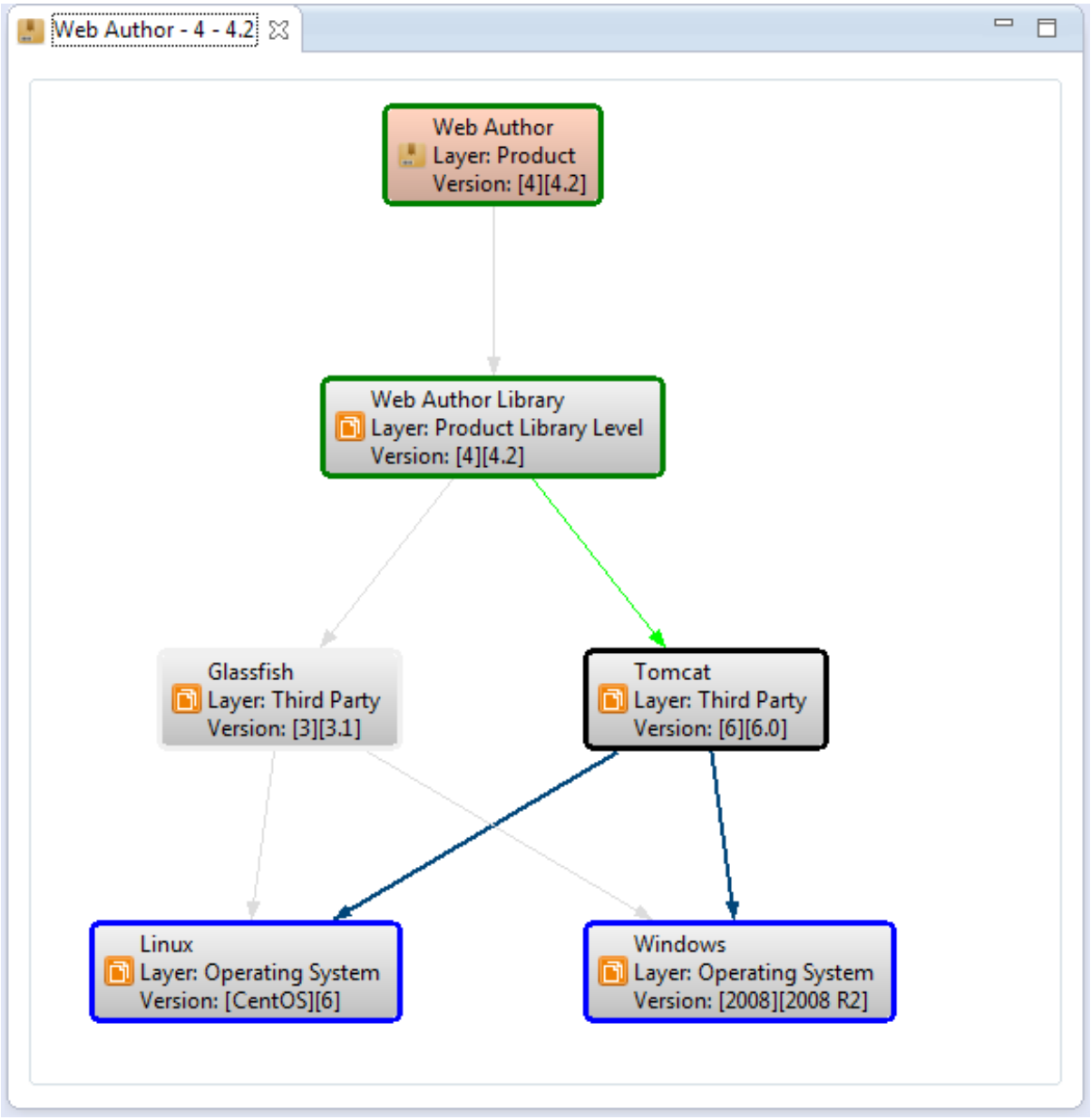
    The changes will be applied automatically once users close and then reopen their DITA CMS. Users can also apply the changes without restarting their DITA CMS by clicking ***DITA CMS*** > ***Synchronize Configuration***.

Working with the Library Dependency Editor

To manage multi-level libraries, you use the Library Dependency Editor, which displays the dependency graph for a library, called the *focus library*.

From the Library Dependency Editor, you can perform the following operations on the focus library: add a child library, remove a child library, and update a child library version.

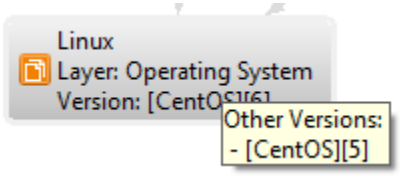
The following diagram shows a sample library dependency graph:



When a diagram is first displayed, the library layers are organized hierarchically, with the highest level on top. The diagram uses the following color codes:

- **Pink**: Library version for which you are seeing dependencies. This is the focus library.
- **Black**: Library version currently selected in the diagram.
- **Green**: Parents of the selected library version.
- **Blue**: Children of the selected library version.

When you hover the mouse over a library version, the other available versions for this library are displayed in a tooltip, as follows:



If no version is available, the message "No other version" is displayed.

Open the Library Dependency Editor

To manage multi-level libraries, you use the Library Dependency Editor.

To open the Library Dependency Editor:

1. **In the Dynamic Release Management view, right-click a library version and select *Open With > Library Dependency Editor*.**

   The Library Dependency Editor is displayed, showing the relationships between the libraries.

2. **To change the display of the Library Dependency Editor:**

   a) **Right-click anywhere in the Library Dependency Editor view and select one of the following options:**

   Table 2: Library Dependency Editor options

Option	Description
Layout style	Specifies how the layers are presented in the Editor. Choices are: • **Compact**: The layers are displayed in a semi-horizontal style, with the highest layer at the top. Layers are condensed, so more than one layer can be displayed on the same horizontal plane. • **Horizontal**: The libraries at the same layer are displayed on the same horizontal plane, with the highest layer at the top of the diagram. • **Vertical**: The libraries at the same layer are displayed on the same vertical plane, with the highest layer at the left of the diagram.
Show Layer	Displays the layer name of the library.

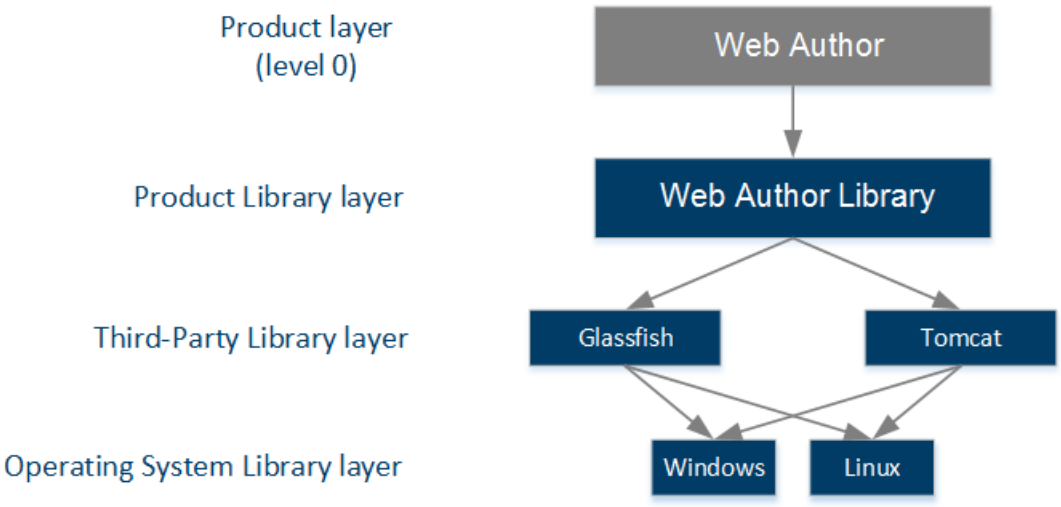
Option	Description
Show Version	Displays the version of the library layer.
Show Dependency	When you select a library layer box, shows the parent libraries in green.
Show Reference	When you select a library layer box, shows the child libraries in blue.

3. **To zoom in or out, hold down the CTRL key and scroll up or down. To reset the zoom, right-click anywhere in the view and select Reset Zoom.**

Creating the library structure of multi-level libraries

To create the library structure in multi-level libraries, you first create the libraries (if they don't already exist) and then use the Library Dependency Editor to set up the dependencies between the libraries.

For example, consider the following library structure:

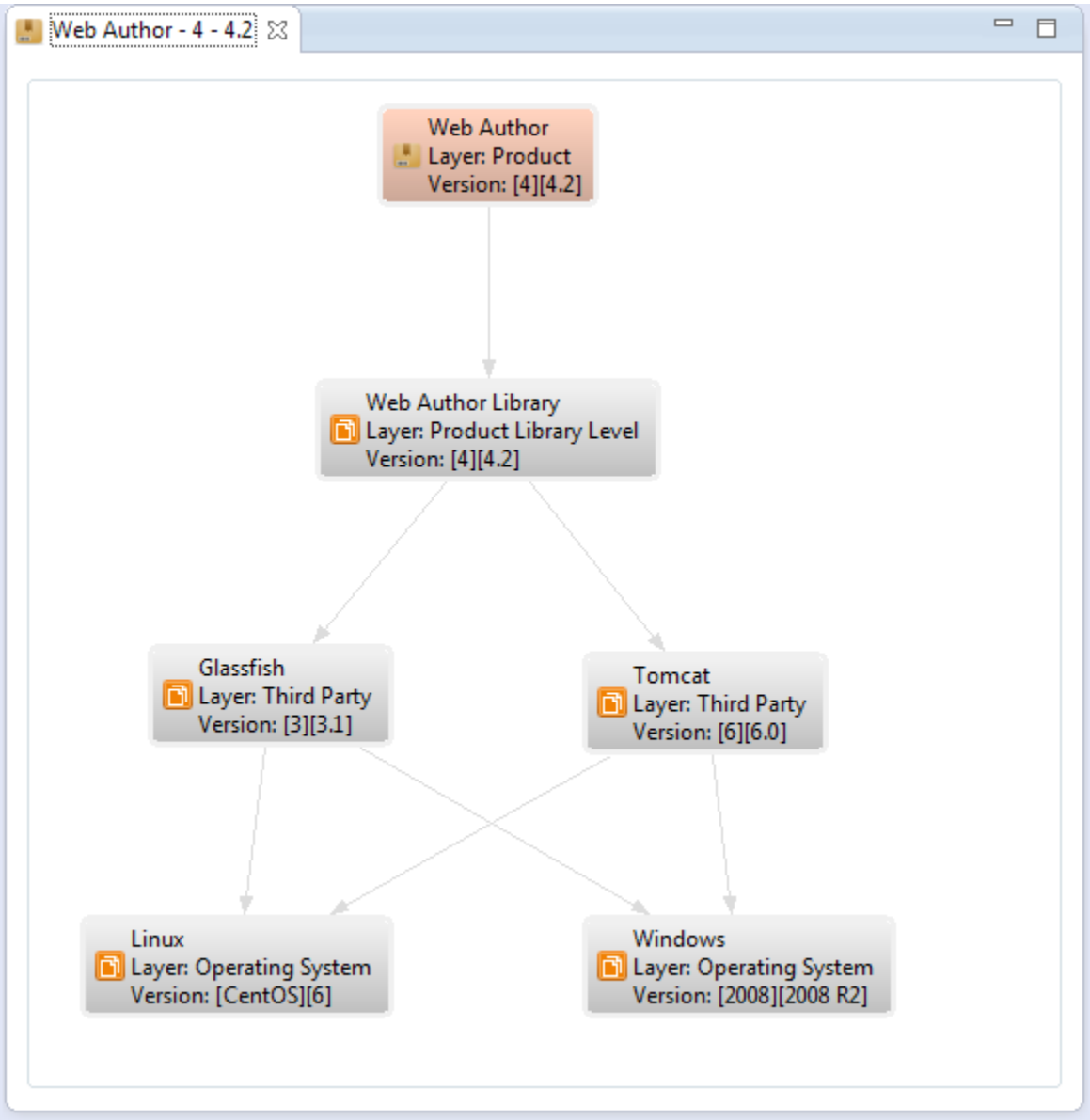


To create this library structure, you:

1. Create each library (Web Author, Tomcat, Glassfish, Windows and Linux) and specify the library level.
2. Create the library releases and versions.
3. Add the Tomcat and Glassfish libraries as children of the Web Author library.
4. Add the Windows and Linux libraries as children of the Tomcat library.

5. Add the Windows and Linux libraries as children of the Glassfish library.

The final structure would look like this in the Library Dependency Editor:



When you add a child library, the children of this library are also added. For example, if you add the Tomcat library as a child of another library, then the Windows and Linux libraries are also added.

Also, as shown above with the Linux and Windows libraries, the same library can be referenced by multiple libraries. However, they must be of the same version in the dependency graph for a library. For example, in the Web Author dependency graph above, you could not add both the CentOS 6 and the CentOS 5 versions of the Linux library. The same library can be included many

times by other libraries in the dependency graph as long as they all use the same version of this library.

Create a new library

Libraries contain topics that can be shared between different products in the DRM.

To create a new library:

1. **In the Dynamic Release Management view, click the Create a new product or library button in the view toolbar.**

   The **Create Product or Library** dialog box is displayed.

2. **Enter the library name.**
3. **In the type field, select Library.**
4. **Specify the language for the library in the Language field.**
5. **If multi-level libraries are enabled in your deployment, select the layer for this library in the Layer field.**

   If the **Layer** field is not available, multi-level libraries are not enabled in your deployment. You can ignore this step.

6. **Enter a description for the library.**
7. **(Optional) Click the Save Settings button to save the Language setting for the next time that you create a library in the Create Product or Library dialog box.**
8. **Click OK.**

   The library is created and displayed in the **Dynamic Release Management** view.

Add child libraries to a focus library

To create a library structure when using multi-level libraries, you add a child library to a focus library.

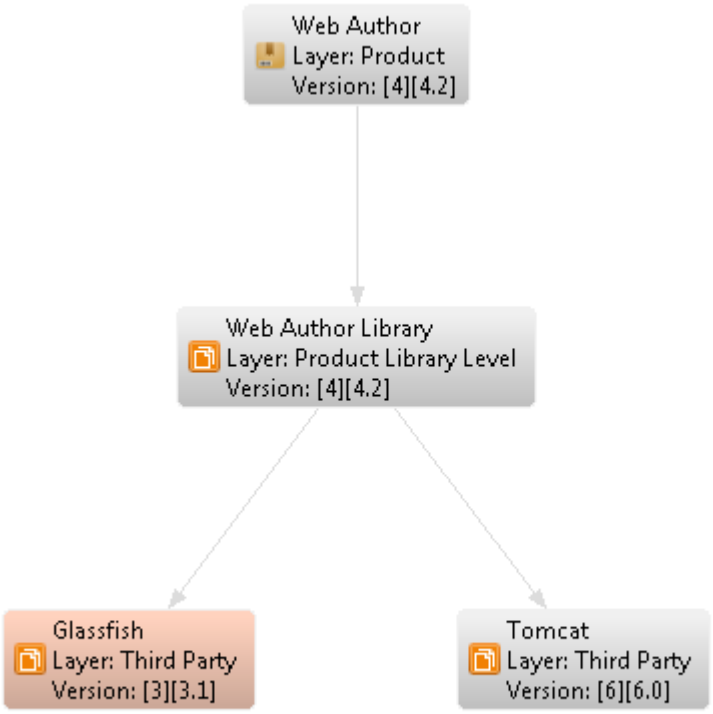
The focus library is the library for which you are seeing dependencies; this library is highlighted in pink.

To add a library as a child of a focus library:

1. **Do one of the following:**

   - In the Dynamic Release Management view, right-click the focus library version and select ***Open With*** > ***Library Dependency Editor***.
   - In the Library Dependency Editor, right-click the library and select **Open With New Editor**.

The current dependencies of the focus library version are displayed in the Library Dependency Editor. The focus library version is displayed in pink. For example:



2. **Right-click the focus library and select Add Library.**

   The Select Versions diagram is displayed.

3. **In the All Libraries pane, select the libraries to add as children of the focus library.**

   Use the filter at the bottom of the pane to filter the libraries in the list by library name.

   The libraries, releases, and versions are displayed in the **Selected Products/Libraries, Releases, and Versions** pane.

4. **In the Selected Products/Libraries, Releases, and Versions pane, select the library versions to add.**

   Use the filter at the bottom of the pane to filter the libraries in the list by release name.

5. **Click OK.**

   The library is added as a child of the focus library and the dependency graph is updated.

Remove a child library

You can remove a child library from a focus library if it is no longer required as a dependency.

To remove a child library from a focus library, the Library Dependency Editor must be open on that focus library.

When you remove a child from a focus library, if this child library also has children, these children will also be removed.

> **Note:** When you remove a library, it is not deleted. It is simply removed as a child of the focus library. If other libraries in the dependency graph are also using this child library, the child library will not be removed from these libraries.
>
> **Tip:** You can only remove a library if:
>
> - Its parent is the focus library, that is, it is displayed in pink. If that is not the case, right-click the parent library, select ***Open With*** > ***Library Dependency Editor***, and try again.
> - The library is not currently referenced by another object; that is, none of its keys are used by one of its parent topics.

1. **Do one of the following:**

   - In the Dynamic Release Management view, right-click the focus library version and select ***Open With*** > ***Library Dependency Editor***.
   - In the Library Dependency Editor, right-click the library and select **Open With New Editor**.

   The current dependencies of the focus library version are displayed in the Library Dependency Editor. The focus library version is displayed in pink.

2. **Right-click the child library to remove and select Remove Library.**

   The library is removed.

Move a library to a different layer

You can move a library to a different layer.

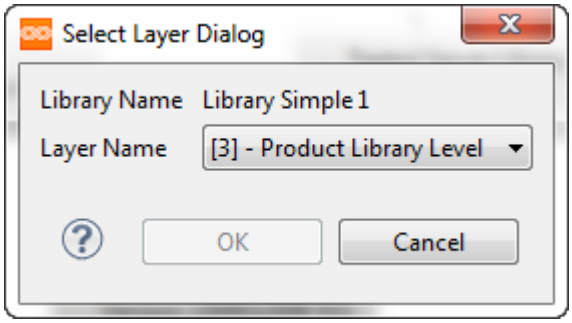
Before moving a library to another layer, the DITA CMS performs the following validation:

- The target layer cannot be lower than the current dependencies of the source layer, but it can be at the same level.
- The target layer cannot be higher than any of the source layer's parents. The Product is not checked, since it cannot have a parent layer (i.e., its layer is 0).

To change the layer level of a library:

1. **Do one of the following:**
   - In the Dynamic Release Management view, right-click the library and select **Change Layer**.
   - In the Library Dependency Editor, right-click the library and select **Change Layer**.

   The Select Layer Dialog is displayed:



2. **In the Library Name drop-down list, select the new layer level for the library.**
3. **Click OK to confirm.**

Update the version of a child library

You can change the version of a child library in a dependency graph.

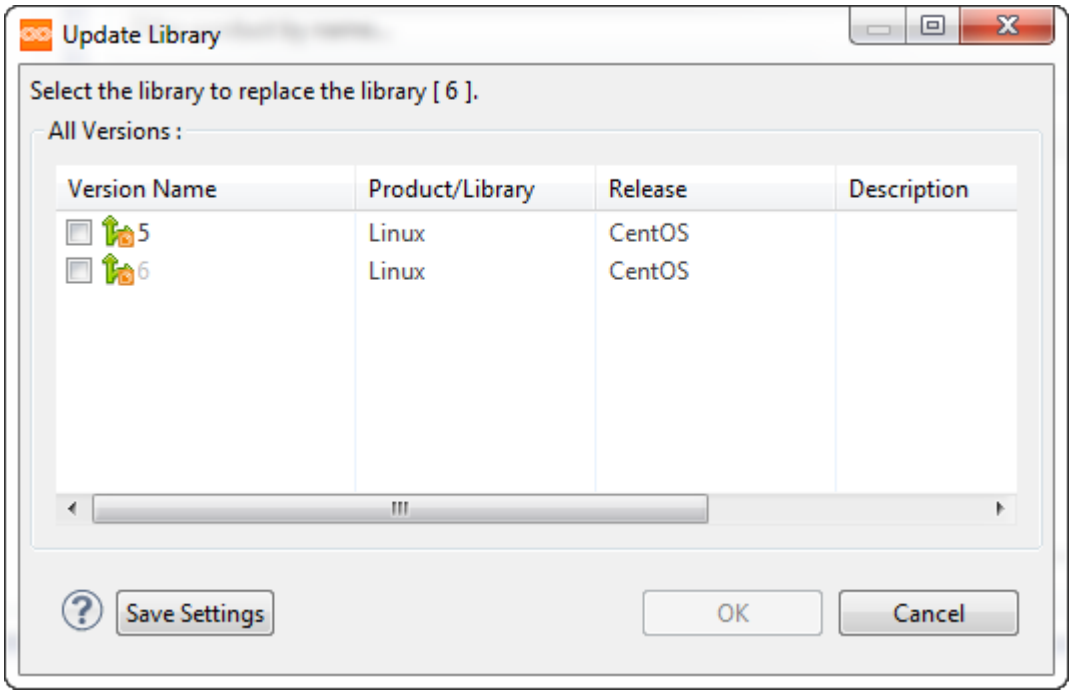
When you change the version of a library, all occurrences of this library in the dependency graph are updated.

To update a library version, the parent of the library must be opened as the focus library in the Library Dependency Editor.

To update an object to another version:

1. **In the Library Dependency Editor, right-click the library to update and select Update Library.**

   The **Update Library** dialog is displayed. It lists the versions for this library:



2. **Select the version and click OK.**

   The library is updated to the selected version.

Summary of right-click menu options in the Library Dependency Editor

The following table summarizes the right-click menu options available on the Library Dependency Editor when a library version is selected:

Table 3: Right-click options

Menu option	
Open with New Editor	Opens a new Library Dependency Editor for the selected library version.
Change Layer	Allows you to change the layer level of a library. See **Move a library to a different layer** on page 35 for the procedure.
Add Library	Adds a child library to the focus. See **Add child libraries to a focus library** on page 33 for the procedure.

Menu option	
Update Library	Updates the version of a child library in a dependency graph. See **Update the version of a child library** on page 36 for the procedure.
Remove Library	Removes a child library from the focus library. See **Remove a child library** on page 35 for the procedure.
Take Screenshot	Takes a screen capture of the Library Dependency Editor view and opens it in the image preview view. You can save the image by clicking the Save icon () button.
Auto Layout	When selected, dynamically organizes the display of libraries when the window is resized.
Pin Selection	Keeps items selected when you click other libraries.
Layout style	Specifies how the layers are presented in the Editor. Choices are: • **Compact**: The layers are displayed in a semi-horizontal style, with the top-most layer at the top. Layers are condensed, so more than one layer can be displayed on the same horizontal plane. • **Horizontal**: The libraries at the same layer are displayed on the same horizontal plane, with the highest layer at the top of the diagram. • **Vertical**: The libraries at the same layer are displayed on the same vertical plane, with the highest layer at the left of the diagram.
Show Layer	Displays the layer name of the library.
Show Version	Displays the version of the library layer.
Show Dependency	When you select a library layer box, shows the parent libraries in green.
Show Reference	When you select a library layer box, shows the child libraries in blue.

Menu option	
Reset Zoom	Resets the zoom to 100%.

Rename a version

You can change a product or library's DRM version name to a new unique name.

For the DRM version to be renamed, all the objects in the selected version must be in a status that allows them to be locked by DITA CMS. If one or more of the objects cannot be locked, the action is cancelled.

To rename the version:

1. **In the Dynamic Release Management view, right-click the version that you want to rename and select Rename Version.**
2. **In the Name box, type the new unique name for the version.**

   The bottom pane of the dialog box displays the version names that already exist in the deployment.
3. **Click OK.**

Miscellaneous improvements to DRM

The DITA CMS 4.3 includes the following miscellaneous improvements to Dynamic Release Management:

- **Locate orphan objects after Update To** - After using ***Dynamic Release Management*** > ***Update to***, some objects may no longer be referenced by any other object in the repository. A list of the orphaned objects is now made available by using the **Locate** button.

3 New Scheduler 4.3 features

Topics:

- Localization Scheduler
- Miscellaneous improvements to Scheduler

This section describes the new features and enhancements introduced in this release of the Scheduler.

Localization Scheduler

The Localization Scheduler allows organizations to automate and customize their localization process.

This feature lets you adapt your process to the requirements of the Translation Management System (TMS) or Localization Service Provider (LSP) vendors that you work with, minimizing the amount of manual operations required on your end.

In particular, this framework allows you to customize the following aspects of the localization process:

- Extracting from the DITA CMS the files to be translated
- Transforming and packaging the files so that they conform to the requirements of your vendors
- Sending the localization package to the vendors
- Retrieving the localized package from the vendors
- Unpackaging and transforming the localized files so that they can be imported into the DITA CMS
- Importing the files back into the DITA CMS

Once these different steps are customized to answer your needs, you can configure the Scheduler so that they are done automatically, at a specified interval.

For more information about this feature, please contact IXIASOFT.

Miscellaneous improvements to Scheduler

Scheduler 4.3 includes the following miscellaneous improvements:

- **Daily reminder notification enhancements in Scheduler** - The Daily Reminder notification now includes the comments provided when the document was assigned, and you can now create a custom XSL stylesheet for the email template and associate it to the DailyReminder job.