XML Documentation for Adobe Experience Manager Installation and Configuration Guide

December 10, 2019

Contents

About this guide

XML Documentation for Adobe Experience Manager (referred to as *XML Documentation solution* later in this guide) is a powerful, enterprise-grade component content management solution (CCMS). It enables native DITA support in Adobe Experience Manager, empowering AEM to handle DITA-based content creation and delivery. It empowers authors to create content using any offline DITA authoring tool, such as Adobe FrameMaker or an easy-to-use built-in Web Editor.

This guide provides the instructions to download, install, and configure XML Documentation solution. In this guide, you will find detailed instructions to set up XML Documentation solution as per your organizational authoring and publishing needs.

This guide is intended for the following type of audiences:

- Administrators, who would install and manage XML Documentation solution on Adobe Experience Manager.
- Publishers, who would run the publishing task to generate output in various formats.

Content structure

The information in this guide is organized as follows:

- *About this guide*: This topic provides an introduction to this guide, intended audience, and how the information is organized in this guide.
- *Download and install*: This topic describes how to download, install, or upgrade XML Documentation solution.
- *User administration and security*: This topic describes the core concept of users and authentication in AEM and the default user groups created by XML Documentation solution.
- *Use custom DITA-OT and DITA specialization*: This topic explains how to configure custom DITA-OT plug-ins and use DITA specialization.
- *Configure document states*: This topic explains how to configure custom states for your DITA documents.
- *Migrate existing content*: This topic describes how to on-board your existing content on AEM repository.
- *Configure topic and map templates*: This topic describes how to configure topic and map templates to meet your authoring needs. Learn about tagging system in AEM and how to configure tags to work with XML Documentation solution. Also, learn how to create your custom snippets.
- *Customize Web Editor*: This topic explains the various customizations that you can make in the Web Editor to enhance its functionality.
- *Enable the Advanced Map Editor feature*: This topic describes that process of enabling the Advanced Map Editor feature.
- *Configure global or folder-level profiles*: This topic explains the process of creating folder profiles and giving permissions to specific users.

- *Version management*: This topic describes how to configure automatic file checkout for files that are opened for editing in the Web Editor.
- *Integrate desktop-based XML editors*: This topic describes the settings you need to enable editing documents in an external XML editor.
- *Configure output generation settings*: This topic describes the various configurations that you can make to customize the default output generation process.
- *Configure and customize workflows*: This topic describes the various configurations to customize the default workflows shipped in XML Documentation solution.
- *Translate content*: This topic provides links to the relevant Help articles in AEM documentation to help you understand and configure the translation framework. Also, learn how to enable component-based translation workflow.
- *Configure DITA content search*: This topic describes how to configure DITA content search in Assets UI and add your custom attributes in search.
- *Appendix*: The appendix contains information about the best practices for working with or setting up features in XML Documentation solution.

Additional resources

Following is a list of other helpful resources of XML Documentation solution, which are available on the Learn & Support page:

- User Guide for Adobe Experience Manager
- API Reference Guide
- Quick Start Guide for Adobe Experience Manager

Download and install

The XML Documentation solution is made available through Adobe Licensing Website (LWS). You can download the XML Documentation solution from your LWS account and install it on all Adobe Experience Manager (AEM) instances in your setup. Typically, your authoring instance and production instance of AEM will be hosted on different servers. You will have to install the XML Documentation solution on all instances of AEM that you intend to use.

Before you begin the download and installation process, you must ensure that your system meets the technical requirements to install the XML Documentation solution.

Technical requirements

Adobe Experience Manager

- Version 6.5 Service Pack 1
- Version 6.4 Service Pack 5
- Version 6.3 Service Pack 3

Operating systems

- Microsoft Windows Server 2012 R2
- Red Hat Linux 7 and 6

Java Development Kit

- Oracle SE 8 JRE 1.8.x
- Oracle SE 7 JRE 1.7.x

Web browser

- Google Chrome

Install Adobe Experience Manager

XML Documentation solution is a plug-in that installs on top of Adobe Experience Manager. Installing AEM requires understanding of some basic AEM concepts and recommended deployment scenarios. The following links will help you get started with AEM installation:

- Basic AEM Concepts
- Recommended AEM Deployments

Once you have identified the deployment strategy that works best for your organization, perform the installation process as described in the *Getting Started* section in AEM documentation.

If you plan to upgrade your AEM instance, then you must follow the given sequence:

1) Uninstall XML Documentation solution.
2) Upgrade your AEM instance.
3) Install XML Documentation solution.

IMPORTANT: There are a number of performance optimization recommendations that you can consider to improve your system performance. See *Recommendations for performance optimization* for details.

Download and install XML Documentation solution for the first time

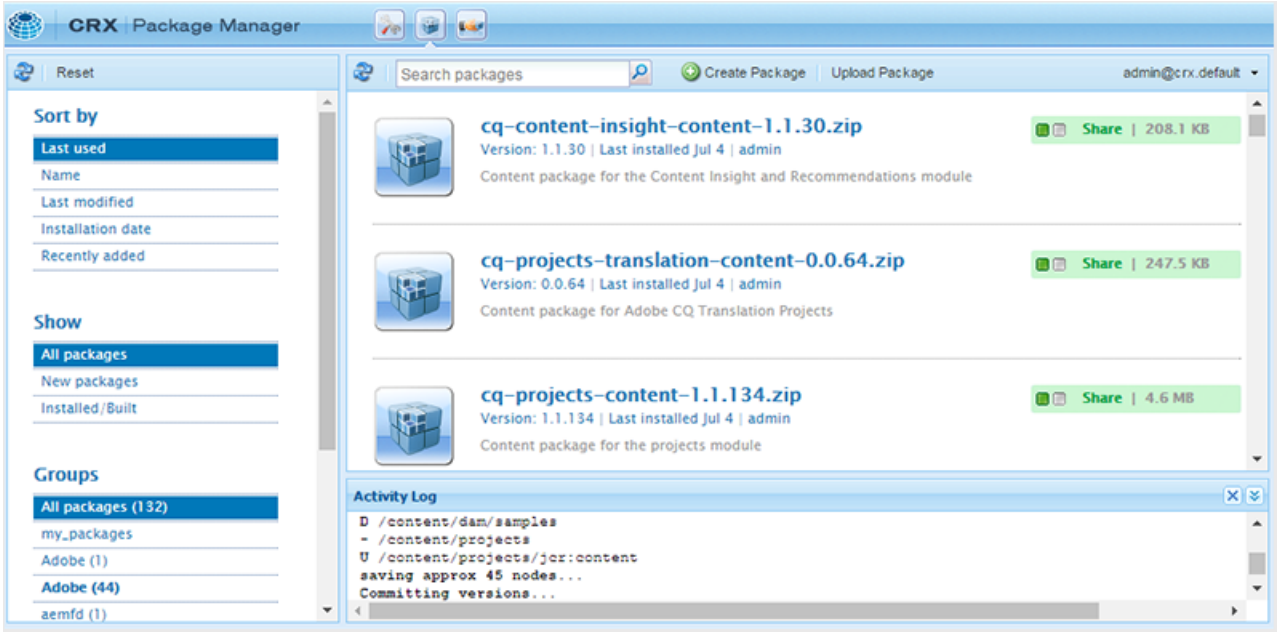
Perform the following steps to download and install XML Documentation solution for the first time on a computer:

IMPORTANT: As XML Documentation solution works in conjunction with AEM's native functionalities, ensure that you have released the lock on all files before starting the installation process. Also, if you want to use Livefyre along with XML Documentation solution, ensure that you install the Livefyre versions earlier than 3.0 before installing XML Documentation solution. If you are using Livefyre version 3.0 or higher, then there is no such restriction.

1) Download the XML Documentation solution from your LWS account.
2) Log into your AEM instance and navigate to the CRX Package Manager. The default URL to access the package manager is:

   `http://<server name>:<port>/crx/packmgr/index.jsp`

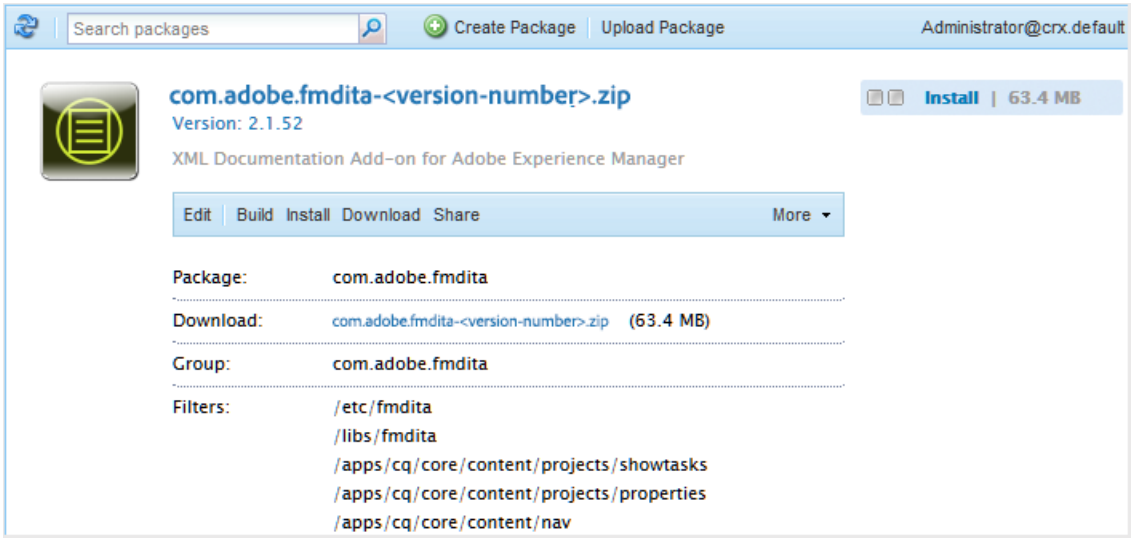
   The Package Manager manages the packages on your local AEM installation. For more information about working with the Package Manager, see How to Work With Packages in AEM documentation.



3) To upload the XML Documentation solution, click **Upload Package**.
4) In the Upload Package dialog, navigate to the XML Documentation solution file that you downloaded in Step 1 and click **OK**.

The package is uploaded to your AEM instance.

5) To install the package, click **Install**.



6) In the Install Package dialog, click **Install**.

7) To get started with the XML Documentation solution, click the Home button  in the upper-left corner of the CRX Package Manager.

NOTE: Perform the installation procedure on all instances of AEM servers in your setup.

Verify XML Documentation solution installation

Once you have installed XML Documentation solution, you need to verify whether the installation was successful or not. Perform the following steps to verify the installation process:

1) Log into your AEM instance and navigate to the AEM Web Console Bundles page. The default URL to access the bundles page is:

   `http://<server name>:<port>/system/console/bundles`

   A list of bundles is shown.

2) Filter the list of bundles by entering `fmdita` in the filtering text box and press **Enter**.

   *The list of bundles is filtered to show the bundles installed by XML Documentation solution. If the installation was successful, then all installed bundles will have a **Status** of **Active**.*

   *If any of the bundles does not have an **Active** status, then check the AEM logs to troubleshoot the installation issue.*

IMPORTANT: There are a number of performance optimization recommendations that you can consider to improve your system performance. See *Recommendations for performance optimization* for details.

Upgrade XML Documentation solution

If you are using XML Documentation solution version 3.2 or later, then you can easily upgrade to the latest version of XML Documentation solution. With the upgrade feature, you don't have to uninstall the previous version of XML Documentation solution. The procedure to upgrade your XML Documentation solution is similar to the fresh install and run a few upgrade scripts. However, before running the process there are certain tasks that you must complete. The following sub-sections will walk you through the prerequisites, configuration and content backup, and running the upgrade scripts.

Upgrading XML Documentation solution is an incremental process, which means that to upgrade from one version to another, you must install all intermediate versions. For example, to upgrade from 3.2 release to 3.5, you must first install 3.3, followed by 3.4, and then finally 3.5.

NOTE: You must get in touch with Adobe Customer Support team if you are upgrading from any version prior to 3.2.

Prerequisites

Before you start the XML Documentation solution upgrade process, ensure that you have:

- Released the lock on all files.
- Imported the review comments in topics open for review.
- Closed all translation tasks.

Backup configurations and baselines

The following table lists the location of various configuration files that you must backup before upgrading XML Documentation solution:

NOTE: You can take a backup of your existing configurations by creating a package using the CRX Package Manager.

Configuration for	Location
DITA profiles	`/content/fmdita/profiles`
Folder profiles	`/content/fmdita/folderprofiles`
Conditional profile	`/content/fmdita/conditionalprofiles`
Map collections	`/content/fmdita/mapcollections`
Document states	`/content/fmdita/states`
Configuration Manager	`/apps/system/config/com.adobe.fmdita.config.ConfigManager.config`
XML Editor	`/apps/system/config/com.adobe.fmdita.xmleditor.config.XmlEditorConfig.config`

The XML Documentation solution upgrade process involves running the upgrade scripts. It is always advisable to take the backup of your content in addition to the server configurations. You can also take

a backup of your baselines, in case the upgrade script fails to migrate your baselines, you can restore them using the backup.

To take a backup of your baselines, perform the following steps:

1) Download and install the baseline upgrade package from the following location:

http://download.adobe.com/pub/adobe/xml-documentation/com.adobe.fmdita.activation-10.0.0.zip

2) Using the Postman tool or simple browser console, create the following request:

```
var json = {
   "activate": false,
   "rules": [

  { "rootPath": "<source_path>", "nodeQueries": [ ".*/metadata/baselines"
] }
   ]}
var ajaxOptions =
{ type: 'post', url: '/bin/fmdita/activate',contentType:
"application/json; charset=UTF8", data: JSON.stringify(json) }
; $.ajax(ajaxOptions);
```

In the above script, replace the `<source_path>` with the folder location of you DITA map file containing the baseline.

3) Build the package created in *Step 2* and download it on your local system.

Download and install the latest version

After taking a backup of your content and configurations, download the latest version of XML Documentation for Adobe Experience Manager. Install the latest version as explained in the *Download and install XML Documentation solution for the first time* section. After installation, run the upgrade scripts as explained in the following section.

Clean up tags folder

You must delete the `rep:policy` node under `/etc/tags` folder from your system. This will result in loss of any customization you might have done on the tag policies.

Run the upgrade scripts

Based on your current version of XML Documentation for Adobe Experience Manager, you need to run specific conversion scripts.

If you are *upgrading from version 3.2 to 3.3*, then you need to run the following baseline upgrade script:

```
http://<server_URL>:<port>/bin/fmdita/baselineModification
?rootPath=<source_path>
```

Replace the `<server_URL>:<port>` with the host and port of your AEM server. The `<source_path>` should be replaced with the folder location of you DITA map file containing the baseline. As this script is system-intensive, ensure that you do not give a generic path. If you give a generic or a path like `/content/dam/`, then the script will not execute successfully. You can run this command multiple times to cover all baselines in multiple DITA maps.

On a successful completion of the script, you will get the name of all updated baselines.

To *move from 3.3 to 3.4*, you need to run the following script to fix the direct references and images references in the baselines:

`http://<server_URL>:<port>/bin/fmdita/baselineModification?rootPath=<source_path>&operation=moveMapDependents`

Replace the `<server_URL>:<port>` with the host and port of your AEM server. The `<source_path>` should be replaced with the folder location of you DITA map file containing the baseline.

On a successful completion of the script, you will get the name of all updated baselines.

Once you have deployed the latest version of XML Documentation solution, restore your content and configurations from the backup.

IMPORTANT: If you are using AEM 6.3, then it is recommended to manually uninstall the older version of XML Documentation solution before upgrading to the latest version. Upgrading without uninstalling the older version might impact the List View in Assets UI. If you have upgraded without uninstalling the older version, then you need to delete the following files:

`/apps/dam/gui/coral/components/admin/contentrenderer/row/asset/link.jsp`

`/apps/dam/gui/coral/components/admin/contentrenderer/row/directory/directory.jsp`

Clear browser cache

After completing the upgrade process, all users must clear the browser cache before using the upgraded version of XML Documentation solution.

Recommendations for performance optimization

This section talks about the performance optimization tasks that you may perform in your setup to achieve optimal performance. The following list covers the type of configurations required to be made, whether is it mandatory or optional, the exact change required in the system, when should you perform the required configuration, and the final result that you achieve by making the change.

- **Configure data store (Mandatory)**

  ***What is the change*?**

  Set the `minRecordLength` property to a value of `100` under the configuration `org.apache.jackrabbit.oak.plugins.blob.datastore.FileDataStore`. For more information about file date store and S3 data store, see the Configuring node stores and data stores in AEM 6 article.

  ***NOTE:** For S3 data store, the cost for maintaining content in data store depends on the number of requests as well. Therefore, when doing this setting with S3—setup cost per request and cache size should also be considered.*

  When to configure?

  After the initial set up, but before any content is migrated. You must perform this change even on an existing system, which ensures that all new content is stored in data store instead of segment store.

Result of this change

The DITA files are saved in data store rather than the segment store. This keeps the segment store size under the recommended limits, which improves the responsiveness of the system.

• Update Lucene index (Mandatory)

***What is the change*?**

Exclude `/content/fmdita` from `oak:index/lucene`.

***NOTE:** XML Documentation solution never uses Lucene indexes to search for content in `/content/fmdita` node.*

When to configure?

If you are making this change on a new system before migrating content, then only updating `oak:index/lucene` is required. Otherwise, on an existing system where the content is already migrated, then after making the change in `oak:index/lucene`, rebuild indexes for Lucene (*which might take few hours to complete*).

Result of this change

This change prevents `/content/fmdita` node from getting indexed and stored in the segment store.

• Java memory optimization (Mandatory)

***What is the change*?**

The JVM start parameters should be carefully tuned based on the infrastructure and the disk size. It is recommended that you consult Adobe Support to get help to access the ideal configuration. You may, however, try out the following configurations yourself:

– Set the JVM heap size to a minimum of 1/4th of the total available memory. Use the parameter `-Xmx<size>` to set the heap memory size.

– Ideally, the size of the heap `dump == xMx`.

– Configure the following parameters to ensure that GC does not run too frequently:

- `-XX:NewRatio`
- `-XX:ParallelGCThreads`
- `-XX:ConcGCThreads`

When to configure?

If you are making this change on an existing system, you need to restart the system. In case of a fresh installation, this change should be made in the start script (`.bat` or `.sh`) file before the system is started.

Result of this change

This results in optimal heap size and regulated execution of GC.

- **Client library minification on Author instance (*Optional*)**

***What is the change*?**

The client libraries should be set to minified in the Authoring instances. This makes sure that there are less bytes to download when a user is browsing the system from different locations. To make this change, set the configuration in **HTML Library Manager** from the Felix console.

When to configure?

This can be done at run time through Felix console or via code deployment.

Result of this change

This change improves the load time of pages on Author instance as less bytes are transferred for loading the client libraries.

- **Configure concurrent publishing threads (*Mandatory, depending on the use case*)**

***What is the change*?**

This change is required if you are using DITA-OT to publish output and a number of concurrent publishing threads is also defined.

By default, XML Documentation solution sets the publishing threads to the number of CPUs+1. However, it is recommended to set this value to half (1/2) or one-third (1/3) of the total number of CPUs. To do this, set the **Generation Pool Size** property under the configuration `com.adobe.fmdita.publish.manager.PublishThreadManagerImpl` as per the recommendations.

When to configure?

This can be done at run time through Felix console or via code deployment.

Result of this change

This change ensures that on a running Author instance all resources are not allocated for the publishing operations. This keeps the system resources available for authors as well, which results in a better user experience.

- **Configure batch size of nodes for AEM Site output generation (Mandatory, depending on the use case)**

***What is the change*?**

This change is required if you are generating AEM Sites output.

Set the **Limit AEM Site Pages in Heap** property under `com.adobe.fmdita.config.ConfigManager` to a number based on your system's configuration. This property defines the batch size of nodes to be committed when the site pages are generated. For example, on a system with a larger number of CPUs and heap size, you can increase the default value from `500` to a larger number. You need to test run with the changed value to come to an optimal value for this property.

When to configure?

This can be done at run time through Felix console or via code deployment.

Result of this change

An increased number of the **Limit AEM Site Pages in Heap** property optimizes the AEM Site output generation process.

- **Patch Xerces Jar while using custom DITA-OT for publishing (Mandatory, depending on the use case)**

***What is the change*?**

This change is required only if you use custom DITA-OT for publishing output.

Replace the Xerces Jar file in your custom DITA-OT package with the one shipped OOTB. The default OOTB `xercesImpl-2.11.0.jar` file is available within the `/etc/fmdita/dita_resources/DITA-OT.zip` file. Ensure that you rename the `xercesImpl-2.11.0.jar` file to match the old Xerces Jar file being replaced.

When to configure?

This can be done at run time.

Result of this change

This change reduces the publishing time and memory utilization while publishing DITA maps with a large number of topics.

- **Optimize number of post-processing threads (Mandatory, depending on the use case)**

***What is the change*?**

This change is required if you are bulk uploading DITA content.

Set the **Post Process Threads** property under `com.adobe.fmdita.config.ConfigManager` to `1`.

When to configure?

This can be done at run time.

Result of this change

This change reduces the post-processing time on bulk upload of DITA files.

User administration and security

To access and configure features in XML Documentation solution, you need to create users. These users can then be assigned permissions to access all or specific features in XML Documentation solution. Learn how to configure and maintain user authorization and also understand the theory behind how authentication and authorization works in AEM.

The following topics in AEM documentation will help you understand the user administration and security related concepts and features:

- Users and Groups in AEM
- Permissions in AEM
- Managing Users and Groups
- Managing Permissions

User groups created by XML Documentation solution

XML Documentation solution provides three out-of-the-box groups to manage different tasks in a DITA project. These groups are: *Authors*, *Reviewers*, and *Publishers*. Depending upon the group a user is associated with, they are allowed to perform specific tasks. For example, publishing task can be performed only by a publisher, but not by an author or a reviewer. Similarly, an author can create a new topic, and a reviewer can only review a topic.

TIP: See *Permissions* for best practices around setting user permissions.

The following table lists various tasks and the groups that can perform those tasks:

Task	Authors	Reviewers	Publishers
Create DITA Topic	Yes		Yes
Create DITA Map	Yes		Yes
Map Collections	Yes		Yes
Create Review Task	Yes		Yes
Review Topic[1]	Yes	Yes	Yes
Key Resolution	Yes		Yes
Open in FrameMaker	Yes		Yes
Check-out/Check-in	Yes		Yes
Edit Topic	Yes		Yes
Move Topic	Yes		Yes

Task	Authors	Reviewers	Publishers
Edit Topic Properties	Yes		Yes
Copy	Yes		Yes
Delete	Yes		Yes
Share	Yes		Yes
Document state			
Create/edit document state profile			Yes
Change document state[2]	Yes	Yes	Yes
Features available in DITA map console (Output Presets tab)			
Generate			Yes
Edit			Yes
Duplicate			Yes
Create			Yes
Delete Preset			Yes
Features available in DITA map console (Outputs tab)			
View generated output	Yes		Yes
Features available in DITA map console (Topics tab)			
Create Review Task	Yes		Yes
Edit	Yes		Yes
Features available in DITA map console (Baselines tab)			
Create			Yes
Edit			Yes
Duplicate			Yes
Remove			Yes
DITA map console (Reports tab)	Yes		Yes
Features available in DITA map console (Condition Presets)			

Task	Authors	Reviewers	Publishers
Create/edit condition preset			Yes

1. If *Authors* and *Publishers* are invited for a review.
2. Depending on the rights given to the user in the document state profile.

Additional notes on user groups

The following list contains some recommendations and points related to user groups and corresponding permissions:

- If you want a user to start the translation or review workflow, ensure that the user is a member of the *Publishers* and *projects-administrators group*.
- Users must have the read, create, delete, and modify permissions available on the source and target language folders that they need to work on.
- If you create a project, you are the owner of the project with *Publishers* permissions. For other users in a project to be able to see their team members, create tasks, or create workflows, they must have read access on `/home/users` and `/home/groups` nodes. One way of giving read access on `/home/users` and `/home/groups` nodes is by giving read access to the `projects-users` group.
- *Reviewers* can access and add review comments on a topic under review from the Project console or from inbox notification link. Also, this access is only available till the time the review task is open.
- By default, *Publishers* are granted access and permissions on the following folders in DAM:
  - `/content/fmdita` –> Read and Write
  - `/content/dam/fmdita-outputs` –> Read and Write
  - `/content/output/sites` –> Read and Write

  You must give explicit read and write permissions to your publisher if you are using any other location apart from the default publishing locations mentioned above.
- All users under *Authors*, *Reviewers*, and *Publishers* groups have read access on all content in DAM.
- Folder-level permissions must be assigned to users via the user administration page.
- If you want your users to be able to perform search operations on DAM, then make your users a member of the *dam-users* group.
- If you want to give admin rights to any user, you can do so by giving them access through AEM standard groups like *administrators*, *projects-administrators*, or OSGI configuration (Felix console).
- To give a user rights to change a document state, make sure that you add the user in the state transition section of the document state profile.

Use custom DITA-OT and DITA specialization

The DITA Open Toolkit (DITA-OT) is a set of Java-based, open source tools that provide processing for DITA maps and topic content. XML Documentation solution allows you to easily import and use custom DITA-OT plug-ins. Once imported, XML Documentation solution can be configured to use the custom DITA-OT plug-in to generate output in any format. At the time of generating the output, simply select the DITA-OT option and the XML Documentation solution uses the custom DITA-OT plug-in to generate the required output.

If you want to process Ant parameters while publishing any output, XML Documentation solution gives you an easy way to do so. You can specify which Ant parameters you want to use and the same are then processed by the publishing process.

NOTE: XML Documentation solution is shipped with DITA-OT version 2.5.2. However, XML Documentation solution supports DITA-OT version 1.7 and above. For the complete list of DITA-OT versions, see DITA-OTversions.

TIP: See *DITA-OT Profiles configuration* and *Using custom DITA-OT* for best practices around using custom DITA-OT plug-ins.

Use custom DITA-OT plug-ins

There are two ways to use custom DITA-OT plug-in for publishing. First method is to upload the custom DITA-OT plug-in into AEM repository. The other method is to save the custom DITA-OT plug-in on your server, create a Profile and provide the location of the custom DITA-OT plug-in in your Profile.

By default, XML Documentation solution comes with a pre-configured Profile that contains the configurations for the default templates to use for editing and publishing content. You can create custom profiles with custom templates to be used while editing documents and custom DITA-OT plug-ins to publish content.

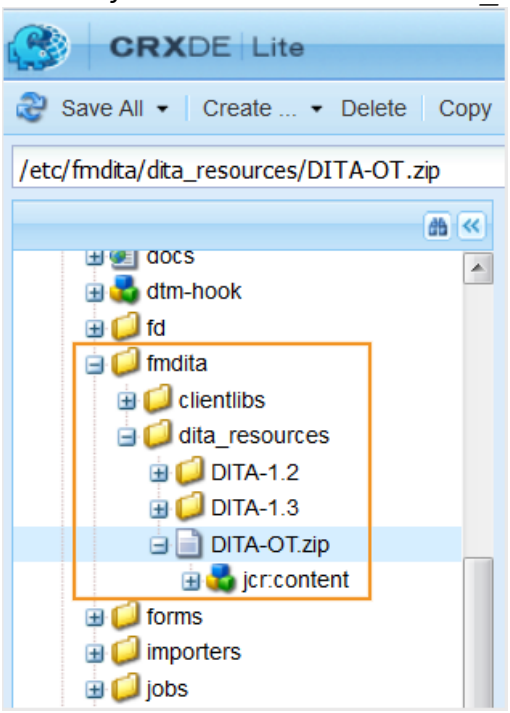
The default DITA-OT package available with XML Documentation solution comes with Apache FOP XSL-FO processor, which does not support rendering of MathML equations. If you are using MathML equations in your content, then ensure that you have integrated a MathML rendering engine plug-in for Apache FOP or use a different XSL-FO processor.

IMPORTANT: If you have upgraded XML Documentation solution from version 2.2 to 2.5.1 or 2.6, then all changes made through the configuration manager are automatically picked and stored in the Default Profile.

Perform the following steps to upload custom DITA-OT plug-in into AEM repository:

1) Log into AEM and open the CRXDE Lite mode.
2) Download the `DITA-OT.ZIP` file.

The location of the `DITA-OT.ZIP` file is `/etc/fmdita/dita_resources/DITA-OT.zip`.



3) Extract the contents of the zip file on your server.
4) Use a DITA-OT plug-in integrator mechanism to integrate the new version of DITA-OT with your custom DITA-OT plug-in.

   NOTE: *For more information about manually integrating plug-ins, see the Manually installing plug-ins topic in DITA-OT documentation.*
5) Create the ZIP file again keeping the same name (`DITA-OT.ZIP`) and the folder structure.
6) Upload the updated ZIP file back into the AEM repository.

   NOTE: *It is recommended not to overwrite the default DITA-OT package. You should upload your custom DITA-OT package containing your plug-in at some other location under the `apps` folder.*
7) Open the default DITA Profile for editing and save it (without making any updates) for the changes to take effect.

Perform the following steps to create a new profile and configure it to use custom DITA-OT plug-in stored on your server:

1) Store the custom DITA-OT plug-in on your server.

   NOTE: *The folder structure for storing the custom DITA-OT plug-in should be: `\<parent-folder>\DITA-OT`.*
2) Click on the Adobe Experience Manager link at the top and choose **Tools**.
3) Select **XML Documentation** from the list of tools.
4) Click on the **DITA Profiles** tile.

   NOTE: The Default Profile information is displayed on the Profiles page. If you have upgraded XML Documentation solution from version 2.2 to 2.5.1 or 2.6, then all changes made through the configuration manager are automatically picked and stored in the Default Profile.

5) You can choose to edit the Default Profile, create a new profile, or duplicate settings from the Default Profile to create a new profile.

NOTE: You can update the Default profile, but you cannot delete it. However, all new profiles that you create can be edited and deleted.

6) Configure the following properties to use the custom DITA-OT plug-in:

Property name	Description
Profile Properties	
Profile Name	Provide a unique name for this profile.
Reuse Output	*(Optional)* If your profile is based on an existing profile, then select this option. Selecting this option ensures that XML Documentation solution does not extract the contents of DITA-OT package again and reuses the existing DITA-OT package.
Profile Extract Path	*(Optional)* Specify the path where DITA-OT is kept on disk. By default, XML Documentation solution bundles a DITA-OT package in its repository and it is extracted on the disk at this path. **NOTE:** You can define this path using any existing system variable or property. See description the *DITA-OT Environment Variables* property for more information.
Assigned Path	(*Optional*) Specify the path in your content repository for which this profile is applicable. You can specify multiple locations.
DITA-OT Properties	
DITA-OT Timeout	(*Optional*) Specify the time (in seconds) for which the XML Documentation solution waits for a response from the DITA-OT plug-in. If no response if received in the specified time, XML Documentation solution terminates the publishing task and the task is flagged as failed. Also, the failure logs are made available in the output generation log file. Default value: 300 seconds (5 minutes)
DITA-OT PDF Arguments	(*Optional*) Specify the custom command-line arguments that are processed by the custom DITA-OT plug-in for generating the PDF output.
DITA-OT AEM Arguments	(*Optional*) Specify the custom command-line arguments that are processed by the custom DITA-OT plug-in for generating the AEM Site output.
DITA-OT Build XML	(*Optional*) Specify the path of the custom Ant build script bundled with the customized DITA-OT plug-in. This path is relative to the DITA-OT directory on your file system.

Property name	Description
DITA-OT Ant Script Folder	(Optional) Specify the path of the DITA-OT Ant script folder. This path is relative to the DITA-OT directory on your file system.
DITA-OT Environment Variables	*(Optional)* Specify environment variables to pass on to the DITA-OT process. By default, the XML Documentation solution adds four variables - `ANT_OPTS`, `ANT_HOME`, `PATH`, and `CLASSPATH`. You can reuse any of the existing system environment variables or properties for building new environment variables. For example, if you have `JAVA_HOME` system variable defined in your system and you want to define a new environment variable called `JAVA_BIN` that is built using `JAVA_HOME`. Then, you can add the definition of `JAVA_BIN` as: `JAVA_BIN= ${JAVA_HOME}/bin` **NOTE:** You can also use Java system properties to build environment variables. For example, if AEM start script defines a Java system property `java.io.tmpdir` to a temporary directory, you can use this property to define new variable as: `${java.io.tmpdir}/fmdita/dita_ot`. **IMPORTANT:** To reuse any existing system variable or property, it must be enclosed within `${}`.
Overwrite DITA-OT Output	*(Optional)* If this option is selected, then you can specify the DITA-OT package available on your local system to generate output using DITA-OT. This configuration is set on activation of the ConfigManager. If you want to specify the path of a DITA-OT package that is stored on AEM server, then deselect this option.
AEM DITA-OT Zip Path/ Local DITA-OT Directory Path	Depending on your selection in the Overwrite DITA-OT Output, specify the complete path where the custom DITA-OT.zip file is stored. This could be the path in your AEM repository or local system.
DITA-OT Plug-in Path	Path of the custom plug-in. This plug-in is integrated automatically with the main DITA-OT package.
Integrate Catalogs	(*Optional*) Path of the custom DTD and XSD catalog.xml files in the AEM repository. This should be provided only when the catalogs are missing from the DITA-OT package. These catalogs are automatically integrated with the main DITA-OT as a plug-in.
Add System ID Catalog	(*Optional*) Select this option only if there are missing Public ID entries in the catalog or if the DITA files use only the System IDs that are relative to the server path from where they are uploaded.

Property name	Description
DITA-OT Temporary Path	*(Optional)* Specify a temporary location where DITA files are copied for processing. Before DITA-OT processes files, they are copied at this temporary location. By default, the temporary storage location is: `<AEM-Install>/crx-quickstart/profiles/ditamaps` **NOTE:** You can define this path using any existing system variable or property. See description the *DITA-OT Environment Variables* property for more information.

NOTE: The XML Documentation solution installer creates two environment variables that you can use to specify the path of the custom DITA-OT plug-in files. These environment variables are: DITAOT_DIR, which contains the path of the DITA-OT directory on the file system; and DITA-MAP_DIR, which contains the path where the DITA map content is extracted on the file system.

7) Click **Done** to save the profile.

NOTE: You can export the custom DITA profile as a package and upload on the other XML Documentation solution instances to save time. For more information, see *Export custom DITA profile as package*.

Integrate DITA specialization

DITA specialization is the process of creating new DITA structures by adding new elements or removing existing elements. To create a new DITA element, you can take an existing DITA element as the base and modify it as per your authoring requirements. In essence, DITA specialization allows you to create customized information models that meet your business requirements while retaining the benefits of the existing DITA architecture.

You can use the Profile feature to store custom DITA specialization settings. These settings can then be used at the time of authoring and publishing custom DITA content. XML Documentation solution allows you to use Public ID and System ID in your custom DTDs/XSDs.

NOTE: XML Documentation solution Web Editor does not have support for XSDs.

Perform the following steps to create a new profile and configure it to use specialized DTDs and XSDs in your XML Documentation solution:

1) Create a specialization folder on your local machine that contains the specialized DTDs and XSDs.
2) Specify the DTD details in the `catalog.xml` file that must also be included in the specialization folder.

   NOTE: *In case of DITA 1.3, the default location for DTD `catalog.xml` file in the AEM repository is: `/etc/fmdita/dita_resources/DITA-1.3/dtd/catalog.xml`.*
3) Specify the XSD details in the `catalog.xml` file that must also be included in the specialization folder.

   NOTE: *In case of DITA 1.3, the default location for XSD catalog.xml file in the AEM repository is: `/etc/fmdita/dita_resources/DITA-1.3/xsd/catalog.xml`.*

4) Upload the folder to the following location:

   `/etc/fmdita/dita_resources`

5) Click on the Adobe Experience Manager link at the top and choose **Tools**.

6) Select **XML Documentation** from the list of tools.

7) Click on the**DITA Profiles** tile.

   NOTE: *The Default Profile information is displayed on the Profiles page. If you have upgraded XML Documentation solution from version 2.2 to 2.5.1 or 2.6, then all changes made through the configuration manager are automatically picked and stored in the Default Profile.*

8) You can choose to edit the Default Profile, create a new profile, or duplicate settings from the Default Profile to create a new profile.

   NOTE: You cannot delete the Default Profile. However, all new profiles that you create can be edited and deleted.

9) In the **Schema** > **Catalog** settings, specify the path of the custom DTD and XSD `catalog.xml` files in your AEM repository.

10) Select the **Add System ID Catalog** option.

    NOTE: Select this option only if there are missing Public ID entries in the catalog or if the DITA files use only the System IDs that are relative to the local file path from where they are uploaded.

    For more information about other properties on the Profiles page, see the properties table in Step 6 of the Use custom DITA-OT plug-ins section.

11) Click **Done** to save the profile.

NOTE: You can export the custom DITA profile as a package and upload on the other XML Documentation solution instances to save time. For more information, see *Export custom DITA profile as package*.

Configure document states

XML Documentation solution lets you define the document states for your DITA topics according to your organization's requirements. You can define different states of your document from start to the end. For example, the first state can be Draft and it can move to Review, Approved, Translated, and finally to Published.

There are two ways in which a topic can transition from one state to another—manual and automatic. The document states that are defined in a profile can be used for manually changing the document state. This can be done from the Properties page of a topic file. Also, you can define who can move the document from one state to another state. For example, an author can create a document and the default state of the document can be Draft. When the author sends the document for review she can change the document state to In-Review. The reviewer can change the document state to either Approved or to Draft again based on the review process. If the document is Approved, the publisher can change the document state to Translated or Published depending on the workflow.

NOTE: If a user belongs to the *administrators* group, the user can change a document's state from any state irrespective of the document state transitions defined in the system.

Create a document state

XML Documentation solution is shipped with a set of default document states. These states are:

- Draft
- Edit
- In-Review
- Approved
- Reviewed
- Done

These default states are available to all DITA topics created under DAM. You can create your own document states and assign these to a specific folder. All DITA files created under that folder will then have access to the newly created document states.

To create document states using the Folder Profile, perform the following steps:

1) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
2) Select XML Documentation from the list of tools.
3) Click on the Document States tile.
   Assets States page is displayed. By default, the page shows a default profile.
4) Click **Create Profile** and enter the following details:
   – Enter the name for the profile in the Profile field.
   – Specify the path where you want to apply the new profile.

- Specify the states of the document in the **Allowed States** under **States**. The default document state are Draft, Edit, In-Review, Approved, and Done.-

  *Click the **Add** button to add a document state.*

  - Click the Delete icon to delete a document state.

  ***NOTE:** Do not delete a document state if documents are still in that state. If you delete a document state, you won't be able to change the document state of such documents unless you belong to the administrator user group.*
- Specify the start state of the document in the **Start State**.
- Specify the end state of the document in the **End State**.
- Specify the state transition of the document in **From** and **To** under **State Transition**.

  *- Specify the users and user groups who can change the document state in **Groups**.*

  *- Click the **Add** button to add a state transition.*

  - Click the Delete icon to delete a state transition.

NOTE: Do not delete a state transition if documents are still in `From` state. If you delete a state transition, you won't be able to change the document state of such documents unless you belong to the *administrator* user group.

5) Click **Done**.

Create a copy of a document state profile

Depending on your requirement, you can create a copy of an existing document state profile. You can use the copy as a base for creating another document profile.

To create a copy of a document state profile, perform the following steps:

1) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
2) Select XML Documentation from the list of tools.
3) Click on the Document States tile.
   Assets States page is displayed.
4) Select the document state profile that you want to duplicate and click **Duplicate Profile**.
5) Make required changes and click **Done**.

Delete a document state or state transition

NOTE: Do not delete a document state or state transition if documents are still in the state or in state transition. If you delete a state or state transition, you won't be able to change the document state of such documents unless you belong to the *administrator* user group.

Perform the following steps to delete a document state or state transition from a document state profile:

1) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*

2) Select XML Documentation from the list of tools.
3) Click on the Document States tile.
   Assets States page is displayed.
4) Select the document state profile from where you want to delete the document state and click **Edit Profile**.
5) Delete the document state or state transition and click **Done**.

Delete a document state profile

To delete a document state profile, perform the following steps:

1) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
2) Select XML Documentation from the list of tools.
3) Click on the Document States tile.
   Assets States page is displayed.
4) Select the document state profile that you want to delete and click **Delete Profile**.

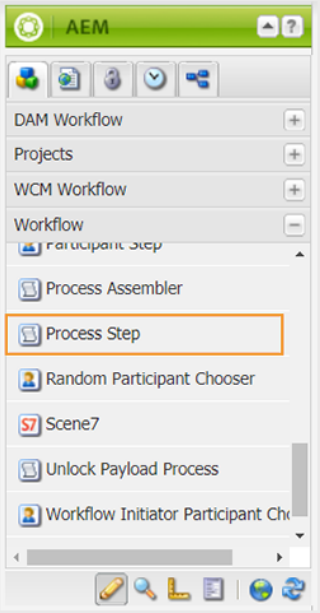
Automate the document state change

If you do not want to manually change the document states, you can create a workflow and automate the document state change.

NOTE: Automated workflows should be in conformance with the document states and transitions defined in the configuration. The system does not perform any checks for state changes done through automated workflows.

Perform the following steps to automate the document state change:

1) Open the workflow page from the AEM server URL.
   <AEM_Server_URL>:<port>/workflow
2) Open a workflow from the workflow page. For example, Review Topic.
3) Select **Process Step** from the **Workflow** section of the AEM dialog and drag-drop on the workflow.



4) Double-click the process and open the **Step Properties** dialog.

5) Enter the following details in the **Process** tab of the dialog and click OK:
   - Select **Set Document State for any DAM asset** from the Process drop-down.
   - Select the Handler Advance check-box.
   - Enter the name of the document state in the **Arguments** text box.

     NOTE: Make sure that you enter the correct document state in the Argument text box. If you enter a wrong name, the document will be set to the wrong document state.

6) Click **Save** to save the workflow.

Migrate existing content

XML Documentation solution allows you to convert a variety of structured and unstructured documents into DITA format. This topic covers the information about uploading your DITA content in AEM repository and converting non-DITA content to DITA format.

TIP: See *Content migration and upload* for best practices around using migrating and uploading existing content.

Upload existing DITA content

Most likely you would have a repository of existing DITA content that you would like to use with the XML Documentation solution. For such existing content, you can use any of the following approaches to bulk upload your content into AEM repository.

Use a WebDAV tool

If you are authoring your topics and maps in any other DITA editor, you can use any WebDAV tool to upload your files. The procedure given in this section uses WinSCP as the WebDAV tool to upload content.

Perform the following steps to use WinSCP to upload files:

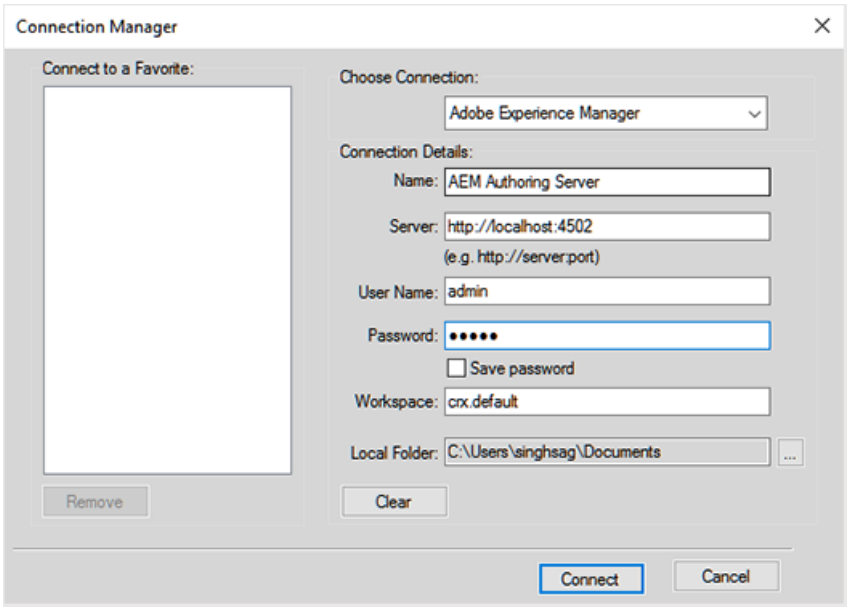
1) Download and install WinSCP on your computer.
2) Launch the WinSCP app.
   The Login dialog appears.
3) On the Login dialog, specify a New Site setting by choosing `WebDAV` as the **File Protocol** and providing other connection details such as:
   - the URL where you AEM server is hosted,
   - the port number (default is 4502), and
   - the user name and password to access your AEM server.
4) Click **Login**.
   On a successful connection, you will see the contents of AEM Assets in the WinSCP user interface. You can easily browse, create, update, or delete content using the WinSCP file explorer.

Use FrameMaker

Adobe FrameMaker comes with a powerful AEM connector that allows you to easily upload your existing DITA and other FrameMaker documents (`.book` and `.fm`) into AEM. You can use various file upload functionalities such as uploading a single file, uploading a complete folder with or without dependencies (like content references, cross-references, and graphics).

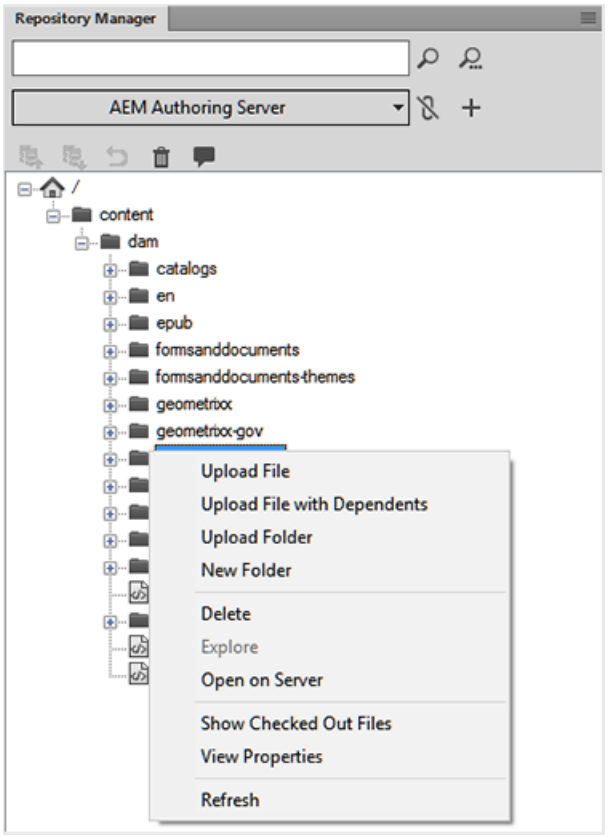
Perform the following steps to use FrameMaker's AEM Connector to upload content:

1) Launch FrameMaker (2017 release).
2) Open the **Connection Manager** dialog.



3) Enter the following details to connect to the AEM repository:
   - **Name**: Enter a descriptive name to identify the connection to your AEM server.
   - **Server**: Enter the URL and port number of your AEM server.
   - **User Name**/**Password**: Enter the user name and password to access AEM server.
4) Click **Connect**.

Once the connection is successfully established, the Assets from AEM repository are displayed in the Repository Manager window.



Right-clicking on any file or folder allows you to perform related operations. For example, if you right-click on a folder, you get options to upload a file, upload file with dependencies, upload an entire folder and so on.

Use curl commands

You can also use curl commands to create a folder in DAM, upload files, and add metadata on the uploaded content.

Create a folder

Run the following command to create a folder in AEM repository:

```
curl --user <username>:<password> --data jcr:primaryType=sling:Folder
"<server folder path>"
```

Specify the following parameters to create a folder:

- `<username>:<passowrd>`: Specify the user name and password to access the AEM repository. This user must have the folder creation privileges.
- `jcr:primaryType=sling:Folder`: Specify this parameter *as is* to create a folder type resource.
- `<server folder path>`: Complete folder path including the name of the new folder that you want to create in AEM repository. For example, if you specify the path as

`http://192.168.1.1:4502/content/dam/projects/xml-documentation-solution`, then the folder `xml-documentation-solution` is created within the `projects` folder in DAM.

Upload a file

Run the following command to upload a file in AEM repository:

```
curl --user <username>:<password> -T "<local file path>" "<server folder path>"
```

Specify the following parameters to upload a file:

- `<username>:<passowrd>`: Specify the user name and password to access the AEM repository. This user must have write privileges on the `server folder path`.
- `local file path`: Complete file path on your local system that you want to upload.
- `<server folder path>`: Complete folder path on the AEM server where you want to upload the file.

Add metadata

Run the following command to add metadata on a file:

```
curl --user <username>:<password> -F<attribute name>=<value> <metadata node path>
```

Specify the following parameters to add metadata information:

- `<username>:<passowrd>`: Specify the user name and password to access the AEM repository. This user must have write privileges on the `metadata node path`.
- `-F<attribute name>=<value>`: The `<attribute name>` is the name of the metadata attribute, such as `audience` and the `<value>` could be `internal`. You can specify multiple attribute name value pairs separated by space.
- `<metadata node path>`: Complete folder path including the file name and its metadata node. For example, if you specify the path as `http://192.168.1.1:4502/content/dam/projects/xml-documentation-solution/intro.xml/jcr:content/metadata`, then the specified metadata information is set on `intro.xml` file.

Migrate non-DITA content

This section guides you through the migration process to migrate non-DITA documents into DITA format. XML Documentation solution provides migration from the following sources:

- *Microsoft Word*
- *InDesign documents*
- *XHTML*
- *DocBook*
- *Unstructured FrameMaker documents*
- *Any other structured document*

Migrate Microsoft Word documents

XML Documentation solution allows you to migrate your existing Word documents (`.docx`) into DITA topic type documents. You need to specify the input and output folder locations along with other parameters and the document gets converted into DITA document. Depending on the content, you could have a `.dita` file and a `.ditamap` file.

To be able to convert a Word document successfully, your document should be well-structured. For example, your document should have a Title, followed by Heading 1, Heading 2, and so on. Each of the headings should have some content in it. If your document is not well-structured, the process might not work as expected.

By default, XML Documentation solution uses the Word-to-DITA (Word2DITA) transformation framework. This transformation depends on the style-to-tag mapping configuration file. To be able to use the Word2DITA transformation successfully, you must consider the following guidelines for preparing your Word document for conversion:

NOTE: If you make any changes in the default style-to-tag mapping configuration file, then you must update and use the guidelines confirming to your updated style mapping.

- Ensure that your document starts with a Title; this Title is mapped to the DITA map title. Also, the Title must be followed by some regular content.
- After the Title, there should be Heading 1, Heading 2, and so on. Each Heading must have some content in it. The Headings are converted into new Concept type topics. The hierarchy of the generated topics is as per the Heading levels in the document, for example, Heading 1 will precede Heading 2, and Heading 2 will precede Heading 3 content.
- The document must have at least one Heading type content.
- Ensure that you do not have any grouped images. In case you have grouped images in your document, ungroup all such images.
- Remove all headers and footers.
- Inline styles such as bold, italics, and underline are converted into **<b>**, **<i>**, and **<u>** elements.
- All ordered and unordered lists are converted into **<ol>** and **<ul>** elements. This also applies to nested lists, lists within tables, notes, or footnotes.
- All hyperlinks are converted into **<xref>**.
- The filename of the converted files is based on the heading text followed by a file number. The file number is a sequential number based on the position of the heading text in the document. For example, if a heading text is “Sample Heading” and it is 10th heading in the document, then the resultant filename for this topic will be similar to `Sample_Heading_10.dita`.

Perform the following steps to convert your existing Word documents into DITA topic type document:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:
   `/libs/fmdita/config/w2d_io.xml`
3) Create an overlay node of the `config` folder within the `apps` node.
4) Navigate to the configuration file available in the `apps` node:

/apps/fmdita/config/w2d_io.xml

The w2d_io.xml file contains the following configurable parameters:

- In the inputDir element, specify the location of the input folder wherein your source Word documents are available. For example, if your Word documents are stored in a folder named wordtodita in projects folder, then specify the location as: /content/dam/projects/wordtodita/
- In theoutputDir element, specify the location of the output folder or keep the default output location to save the converted DITA document. If the specified output folder does not exist on DAM, then the conversion workflow creates the output folder.
- For the createRev element, specify whether a new version of the converted DITA topic is to be created (true) or not (false).
- In the s2tMap element, specify the location of the map file that contains mappings for Word document styles to DITA elements. The default mapping is stored in the file located at:

  /etc/fmdita/word2dita/word-builtin-styles-style2tagmap.xml

  NOTE: *For more information about the structure of word-builtin-styles-style2tagmap.xml file and how you can customize it, see Style to Tag Mapping in DITA For Publishers User Guide.*

5) Save the w2d_io.xml file.
6) After configuring the required parameters in the w2d_io.xml file, log into AEM and open the Assets UI.
7) Navigate to the input folder location (wordtodita).
8) Upload the source Word documents into this folder. For information on uploading content on DAM, see *Upload existing DITA content*.

Using the **<config> </config>** block, you can define one or multiple blocks of configurations for conversion. The conversion workflow gets executed and the final output in the form of a DITA topic is saved in the location specified in the **<outputDir>** element.

Migrate Adobe InDesign documents

XML Documentation solution allows you to convert InDesign documents. Similar to FrameMaker, InDesign also allows you to create unstructured and structured documents. The unstructured documents use the paragraph and character styles to format content. The structured document use elements and their corresponding attributes.

The conversion process requires mapping the paragraph and character style formats to relevant DITA elements. Similarly, in case of structured documents, the mapping file will contain one-to-one mapping of InDesign elements and attributes with DITA elements and attributes.

The conversion process involves the following actions in the backend:

- The *InDesign Markup Language* (IDML) file is unpacked to a working directory.
- The designmap.xml file is read to locate the individual InDesign stories.
- All stories are merged into a single XML instance, 'empty' stories are discarded.
- All embedded graphics are exported.

- Pre-conversion of standard structures such as tables and graphics into DITA format.
- Style or structure mapping to final output based on the mapping file.
- Creation and validation of individual DITA topics and DITA map files.
- Deletion of temporary files.

Broadly, the conversion process requires you to *Prepare InDesign files for conversion* and *Prepare the mapping file for InDesign to DITA migration*, then you need to follow the given procedure of running the conversion process.

Perform the following steps to convert your existing InDesign documents into DITA topic type document:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:
   `/libs/fmdita/config/idml2dita_io.xml`
3) Create an overlay node of the `config` folder within the `apps` node.
4) Navigate to the configuration file available in the `apps` node:
   `/apps/fmdita/config/idml2dita_io.xml`
   Configure the following parameters in the `idml2dita_io.xml` file:
   - In the `inputDir` element, specify the location of the input folder wherein your source InDesign documents are available. For example, if your InDesign documents are stored in a folder named `indesigntodita` in `projects` folder, then specify the location as: `/content/dam/idmlfiles/indesigntodita/`
   - In the`outputDir` element, specify the location of the output folder or keep the default output location to save the converted DITA document. If the specified output folder does not exist on DAM, then the conversion workflow creates the output folder.
   - In the `mapStyle` element, specify the location of the map file that contains mappings for InDesign document styles to DITA elements. The default mapping is stored in the file located at:
     `/stmap.adobeidml.xml`
     ***NOTE:** For more information about the structure of `stmap.adobeidml.xml` file and how you can customize it, see the section Prepare the mapping file for InDesign to DITA migration in Appendix.*
5) Save the `idml2dita_io.xml` file.
6) After configuring the required parameters in the `idml2dita_io.xml` file, log into AEM and open the Assets UI.
7) Navigate to the input folder location (`indesigntodita`).
8) Upload the source InDesign documents into this folder. For information on uploading content on DAM, see *Upload existing DITA content*.

Migrate XHTML documents

XML Documentation solution allows you to convert your existing XHTML documents into DITA topic type documents. You need to specify the input and output folder locations along with other parameters and

the documents get converted into DITA format. There are two methods that you can use to convert your structured HTML documents:

- Upload all documents into the input folder, or
- Create a ZIP of all documents along with the media files and upload it into the input folder. This approach is generally used for a set of HTML files that are linked to each other and there is a table of contents (`index.html`). The `index.html` file contains links to all HTML files in the set.

Whether you upload all files individually or bundled in a ZIP, the conversion process creates a one-to-one mapping between the HTML files and the resultant DITA files. This essentially means that there is a `.dita` file created for each `.html` file in the input folder.

The following points must be considered for uploading your documents in a ZIP file:

- All referenced topics should be placed within the ZIP file.
- All referenced media files should be referred within the topic files using relative link.
- Create an `index.html` file and add links to the topics that you want to add in the TOC. This `index.html` file is used to create the DITA map file. In the `index.html` file, you can also create nested topics listing as shown in the following code sample:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<html
xmlns=”http://www.w3.org/1999/xhtml”>
    <head>
        <title>Sample Index File</title>
    </head>
    <body>
        <h1>Sample Index</h1>
        <div class=”content”>
            <ul class=”book”>
                <li class=”topicref”>
                    <a href=”Topic1.html”>Topic 1</a>
                    <ul class=”book”>
                        <li class=”topicref”>
                            <a href=”Topic1-1.html”>Topic 1.1</a>
                        </li>
                        <li class=”topicref”>
                            <a href=”Topic1-2.html”>Topic 1.2</a>
                        </li>
                    </ul>
                </li>
                <li class=”topicref”>
                    <a href=”Topic2.html”>Topic 2</a>
                </li>
            </ul>
        </div>
    </body>
</html>
```

Note that every **<ul>** *tag must have the* `class` *attribute set to* `book`*. Similarly, every* **<li>** *tag's* `class` *must be set to* `topicref`*.*

- If you use inline styles, then convert the inline styles to CSS-based style classes in your XHTML file. Then use style-attribute mapping to convert these classes based styles to DITA `outputclass` attribute in the converted DITA file.

  While generating HTML or AEM Site output from these DITA files, this `outputclass` attributes can be used to apply style-class on generated HTML or AEM Site to match your source HTML content.

Apart from the considerations for creating the ZIP file, your XHTML document must also be well-structured. For example, your document should have a *Title*, followed by *Heading 1*, *Heading 2*, and so on. Each of the headings should have some content in it. If your document is not well-structured, the migration process might not work as expected.

To convert your existing XHTML document into DITA topic, perform the following steps:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:

   /libs/fmdita/config/h2d_io.xml
3) Create an overlay node of the `config` folder within the `apps` node.
4) Navigate to the configuration file available in the `apps` node:

   /apps/fmdita/config/h2d_io.xml

   The `h2d_io.xml` file contains the following configurable parameters:
   - In the `inputDir` element, specify the location of your input folder wherein your source XHTML documents are available. For example, if your XHTML documents are stored in a folder named `xhtmltodita` in `projects` folder, then specify the location as: `/content/dam/projects/xhtmltodita/`
   - In the`outputDir` element, specify the location of your output folder or keep the default output location. If the specified output folder does not exist on DAM, then the conversion workflow creates the output folder.
   - For the `createRev` element, specify whether a new version of the converted DITA topic is to be created (`true`) or not (`false`).
5) Save the `h2d_io.xml` file.
6) After configuring the required parameters in the `w2d_io.xml` file, log into AEM and open the Assets UI.
7) *(Optional)* You can also add the related links section to the converted documents. Perform the following steps to enable this feature:

   NOTE: By default, the related links section is not created in the converted documents.
   a) Navigate to the `h2d.xsl` file in the following location:

      /etc/fmdita/html2dita/
   b) Search for the following parameter:

      <xsl:param name="generate-related-links" select="false()"/>
   c) Set the value of the above parameter to `true()`.
   d) Save and close the file.
8) Navigate to the input folder location (`xhtmltodita`).

9) Upload the source XHTML documents into this folder. For information on uploading content on DAM, see *Upload existing DITA content*.

Using the `<config> </config>` block, you can define one or multiple blocks of configurations for conversion. The conversion workflow gets executed and the final output in the form of a DITA topics is saved in the location specified in the `outputDir` element.

Migrate DocBooks

XML Documentation solution allows you to convert your existing DocBook documents into DITA documents. You need to specify the input and output folder locations, the files that need to be processed, and whether a new version of the document is required or not. Depending on the content, you could have a `.dita` file and a `.ditamap` file.

To convert your existing DocBook documents into DITA format, perform the following steps:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:
   /libs/fmdita/config/doc2dita_io.xml
3) Create an overlay node of the `config` folder within the `apps` node.
4) Navigate to the configuration file available in the `apps` node:
   /apps/fmdita/config/doc2dita_io.xml
   The `doc2dita_io.xml` file contains the following configurable parameters:
   - In the `inputDir` element, specify the location of your input folder wherein your source DocBook documents are available. For example, if your DocBook documents are stored in a folder named `docbooktodita` in `projects` folder, then specify the location as: `/content/dam/projects/docbooktodita/`
   - In the`outputDir` element, specify the location of your output folder or keep the default output location. If the specified output folder does not exist on DAM, then the conversion workflow creates the output folder.
   - In the `inpExt` element, specify the file extensions that need to be picked up for conversion. By default, the XML and DocBook file extensions are picked up for conversion. If your files are saved in any other extension, you can specify those file extensions in this parameter. If you have multiple file extensions, then specify those file extensions separated by a comma.
   - In the `assetExt` element, specify the file extensions of the image files that need to be picked up for processing. By default, the JPG, GIF, EPS, and PNG file extensions are picked up for processing. If your files are saved in any other extension, you can specify those file extensions in this parameter. If you have multiple file extensions, then specify those file extensions separated by a comma.
   - For the `createRev` element, specify whether a new version of the converted DITA topic is to be created (`true`) or not (`false`).
5) Save the `doc2dita_io.xml` file.
6) After configuring the required parameters in the `doc2dita_io.xml` file, log into AEM and open the Assets UI.

7) Navigate to the input folder location (`docbooktodita`).
8) Upload the source DocBook documents into this folder. For information on uploading content on DAM, see *Upload existing DITA content*.

Using the `<config> </config>` block, you can define one or multiple blocks of configurations for conversion. The conversion workflow gets executed and the final output in the form of a DITA topic is saved in the location specified in the `outputDir` element.

Migrate unstructured FrameMaker documents

XML Documentation solution allows you to convert your existing unstructured FrameMaker (`.fm` and `.book`) documents into DITA documents. The first step is to create style mappings using FrameMaker and save those settings in a `.sts` file. Next, if you are using custom DITA, then you can map your custom elements with the source FrameMaker formats in the `ditaElems.xml` file. For example, if you have created a custom element named `impnote` to handle all important notes, then you can define this custom element in the `ditaElems.xml` file. Once this custom element is defined, XML Documentation solution would not raise an error while converting FrameMaker document containing `impnote` element.

Also, If you want to specify some additional attributes with your custom or valid DITA element, you can define those in the `style2attrMap.xml` file. For example, you can specify the `type` attribute with the value of `important` to be passed on with the `impnote` element. This additional information can be specified in the `style2attrMap.xml` file.

In addition to specifying

To convert your existing unstructured FrameMaker documents into DITA format, perform the following steps:

1) Create style mappings in FrameMaker and save those settings in a `.sts` file.
2) Log into AEM and open the CRXDE Lite mode.
3) If you have custom DITA elements, define those in the `ditaElems.xml` file available at the following location:

   `/libs/fmdita/config/ditaElems.xml`
4) Create an overlay node of the `config` folder within the `apps` node.
5) Navigate to the configuration file available in the `apps` node:

   `/apps/fmdita/config/ditaElems.xml`

   The `ditaElems.xml` file contains a single configurable parameter:
   - In the `elem` parameter, specify the name of the custom element that you want to use in your converted DITA documents. This element would be passed on as is in the generated DITA documents.
6) If you want to specify additional attributes, define those in the `style2attrMap.xml` file available at the following location:

   `/libs/fmdita/config/style2attrMap.xml`
7) Create an overlay node of the `config` folder within the `apps` node.
8) Navigate to the configuration file available in the `apps` node:

/apps/fmdita/config/style2attrMap.xml

The style2attrMap.xml file contains the following configurable parameters:

- In the fmStyle parameter, specify the source format used in the FrameMaker document that you want to map.
- In theditaAttr element, specify the DITA attribute that you want to map with the source format.
- In the ditaVal element, specify the value for the mapped attribute. If you don't have any value, you can leave this entry blank.

9) Save the style2attrMap.xml file.
10) After configuring the required parameters in the style2attrMap.xml file, log into AEM and open the Assets UI.
11) Navigate to and click on the FrameMaker document that you want to convert.

    The DITA map console appears showing the list of Output Presets available to generate output.
12) Select DITA output format and configure the required parameters.

    ***NOTE:** You must use the same settings file (.sts) that you created in FrameMaker. Also, specify the Settings Name and Destination Path.*
13) Click the **Generate** icon to start the output generation process.

Using the <attrMap> </attrMap> block, you can define one or multiple blocks of configurations for conversion. Depending on the content, you could have a .dita file and a .ditamap file as the converted files.

Migrate any other structured document

XML Documentation solution allows you to convert your existing structured documents into valid DITA documents. You need to specify the input and output folder locations, the location of your transformation file, the extension with which the final output is saved, and whether a new version of the document is required or not.

To convert your existing structured documents into DITA format, perform the following steps:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:

   /libs/fmdita/config/XSLConfig.xml
3) Create an overlay node of the config folder within the apps node.
4) Navigate to the configuration file available in the apps node:

   /apps/fmdita/config/XSLConfig.xml

   The XSLConfig.xml file contains the following configurable parameters:

   - In the inputDir element, specify the location of your input folder wherein your source structured documents are available. For example, if your structured documents are stored in a folder named xsltodita in projects folder, then specify the location as: /content/dam/projects/xsltodita/

- In the`outputDir` element, specify the location of your output folder or keep the default output location. If the specified output folder does not exist on DAM, then the conversion workflow creates the output folder.
- In the `xslFolder` element, specify the location of the folder where the XSL transformation files are stored.
- In the `xslPath` element, specify the location of the primary `.XSL` file that is used to initiate the conversion process.
- In the `outputExt` element, specify the file extensions of the final output file that is created from the transformation stream.
- For the `createRev` element, specify whether a new version of the converted DITA topic is to be created (`true`) or not (`false`).

5) Save the `XSLConfig.xml` file.
6) After configuring the required parameters in the `XSLConfig.xml` file, log into AEM and open the Assets UI.
7) Navigate to the input folder location (`xsltodita`).
8) Upload the source structured documents into this folder. For information on uploading content on DAM, see *Upload existing DITA content*.

Using the `<config> </config>` block, you can define one or multiple blocks of configurations for conversion. The conversion workflow gets executed and the final output in the form of a DITA topic is saved in the location specified in the `outputDir` element.

Configure topic and map templates

XML Documentation solution comes with topic and DITA map templates. You can create your own custom template and share the same with your authors. This topic covers the process of using custom topic and map templates.

Configure custom DITA topic template

The XML Documentation solution comes with the following DITA topic templates:

- Topic
- Task
- Concept
- Reference
- Glossary
- Troubleshooting
- Blank

You can use any of these templates to create topics templates as per your authoring requirements. The Blank DITA template does not contain any structure or elements like the other templates. You can use the Blank template as the base, if your template is highly customized and is not based on any regular DITA topic templates.

To customize DITA topic template and use it for authoring, you need to perform the following three main tasks:

1) *(Optional)* *Configure custom DITA template folder path*
2) *Create custom authoring template*
3) Add a custom template into the global or folder-level profile as explained in the *Configure authoring templates* section

Configure custom DITA template folder path

XML Documentation solution allows you to configure a folder to store your customized DITA map and templates. By default, the template files are stored in the following folder in DAM:

`/content/dam/dita-templates/`

To manage topic and map template files, there are dedicated folders to store the topic and map templates. By default, all topic templates are stored under the `/content/dam/dita-templates/topics`

folder. All map templates are stored under the `/content/dam/dita-templates/maps` folder.

As an administrator, you can choose to create custom map or topic templates in the default folder or create your own folder to store custom templates. If you plan to use the default folder, then you can skip this process.

To configure a folder for your custom DITA topic templates, perform the following steps:

IMPORTANT: You can skip this process if you want to use the default folder to store custom templates.

1) Open the Adobe Experience Manager Web Console Configuration page.

   The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

2) Search for and click on the *com.adobe.fmdita.config.ConfigManager* bundle.
3) In the **Templates Location** property, specify a location to store custom templates.
4) Click **Save**.

If the specified location exists in DAM, then all default map and topic templates are copied into that folder. If the location does not exist, then the folder is created with all default map and topic templates.

Configure custom DITA map template

XML Documentation solution comes with two out-of-the-box map templates — DITA map and Bookmap. You can create maps based on these templates; or, you can define your own map templates that can then be used to create new maps.

Perform the following steps to add your custom map templates:

1) Log into Adobe Experience Manager as an administrator.
2) In the Assets UI, navigate to the folder configured to store the map template files. By default, all map templates are stored in the `/content/dam/dita-templates/maps` folder.

   NOTE: To configure a custom location to store topic or map templates, see *Configure custom DITA template folder path*
3) Click **Create** > **DITA Template**.
4) On the Blueprint page, select the type of the map template that you want to create.

   NOTE: You can use the Map or Bookmap template as the base for your new template.
5) Click **Next**.
6) On the new template Properties page, enter a **Title** and **Name** for the template.

   NOTE: *The name is automatically suggested based on the Title of your template. If you want to manually specify the name, then ensure that the Name does not contain any spaces, apostrophe, or braces and ends with* `.ditamap`*.*
7) Click **Create**.

   The Map Created message appears.

   You can choose to open the template for editing in the Map Editor, or save the template file in the template store location. Once the template is created, you can use the Map Editor to customize the template as per your authoring needs. Once a template is in place, ensure that you associate it either with a global or folder-level profile.

Next time you create a new map, your template shows up in the Blueprint page. For more information about creating a DITA map, see the XML Documentation for Adobe Experience Manager User Guide.

TIP: See *Custom templates* for best practices around using custom map templates.

Configure snippets

A snippet is a static piece of content that can be reused across documents. As an administrator, you might require your authors to use a standard text and image combination for a particular type of element. One approach is to share the guidelines with authors to create element as per a specific rule, which is prone to errors. Alternatively, you can create snippets with predefined text and images, and share the same with your authors to use.

XML Documentation solution allows you to configure reusable snippets that you can share with your authors to use in their documents. The snippets information can be stored in the form of a JSON file, which can be stored anywhere in your DAM. Then, you need to configure the file path in the *configMgr* and it is made available to authors within any project.

Perform the following steps to add a snippet:

1) Create a JSON file containing the key and value pair. The following code snippet defines two snippets – one for a *Warning* type message and other for an *Error*.

```
{ "Warning" : "<p>This is a warning:</p>", "Error" : "<p>
   <image href='/sample/Images2/error_icon.png'/>
   <b>This is an error:</b>
</p>" }
```

The Warning snippet contains only the predefined text. However, the Error snippet inserts an image along with the predefined text. Also, the text is formatted using the b (bold) tag.

2) Save the JSON file in a DAM location, for example - `/apps/fmdita/config/note_snippet.json`.

3) Open the Adobe Experience Manager Web Console Configuration page. The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

   Search for and click on the com.adobe.fmdita.xmleditor.config.XmlEditorConfig bundle.

4) Specify the absolute path of the JSON file in the **Snippets Config Path** field.

   This should be same path as specified in Step 2:

   `/apps/fmdita/config/note_snippet.json`

5) Click **Save**.

Your authors will now get to see and use two snippets — *Warning* and *Error*.

Customize Web Editor

XML Documentation solution comes bundled with a powerful Web Editor that allows your authors to create and edit DITA documents. You can customize the Web Editor's toolbar to add or remove any feature that you can access from the toolbar. Also, you can change the look-and-feel of the document when it is opened in the Web Editor.

Customize toolbar

By default, the Web Editor is shipped with the most common editorial features required by any DITA editor. Features such as inserting elements of type list (numbered or bulleted), cross-reference, content reference, table, paragraph, and character formatting are available in the editor. In addition to these basic elements, you can customize the Web Editor to insert elements that are used in your authoring environment.

There are two ways of customizing the Web Editor's toolbar:

- Add a new functionality to the toolbar
- Remove any existing functionality from the toolbar

Add a feature in the toolbar

Adding a functionality to the Web Editor involves two primary tasks - adding an icon for the feature in the *ui_config.json* file and adding the background functionality in JavaScript.

Perform the following steps to add a feature to the Web Editor's toolbar:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:
   `/ibs/fmdita/clientlibs/clientlibs/xmleditor/ui_config.json`
3) Create a copy of the default configuration file at the following location:
   `/apps/fmdita/xmleditor/ui_config.json`
4) Navigate to and open the `ui_config.json` file in the `apps` node for editing.
5) In the `ui_config.json` file, add the definition of the new feature in the toolbars section. Typically, you can create a new toolbar button group and add one or more toolbar buttons to it. Or, you can add a new toolbar button within an existing toolbar group. The following details are required to create a new toolbar group:

type

Specify `blockGroup` as the `type` value. This value indicates that you are creating a block group that would contain one or more toolbar groups.

extraclass

Name of the class or classes separated with space.

items

Specify the definition of all groups in the toolbar. Each group can contain one or multiple toolbar icons. To define icons within a toolbar group, you need to again define the `type` attribute within the `items`, and set its value to `buttonGroup`. Specify one or more class names in the `extraclass` property. Specify the feature name in the `label` property. The following snippet from the `ui_config.json` file shows the definition for the main toolbar block, followed by the `buttonGroup` definition:

```
"toolbar": {
"type": "blockGroup",
"extraclass":
"toolbar operations",
"items": [
{
"type": "buttonGroup",
"extraclass": "left-controls",
"label": "Left Controls",
"items": [
```

Within the `items` collection, you need to specify the definition for one or more toolbar icons.

You need to define the following properties to add a toolbar icon:

type

Specify `button` as the `type` value. This value indicates that you are adding a toolbar button.

icon

Specify the name of the Coral icon that you want to use in the toolbar.

variant

Specify `quiet` as the `variant` value.

title

Specify the tooltip for the icon.

on-click

Specify the command name defined for the feature in the JavaScript file. If your command requires input parameters, then specify the command name as:

```
"on-click": {"name": "AUTHOR_INSERT_ELEMENT", "args": "simpletable"}
```

show or hide

If you are defining the `show` property, then specify the modes in which the icon is displayed. Possible values are - `@isAuthorMode`, `@isSourceMode`, `@isPreviewMode`, `true` (display in all modes), or `false` (hide in all modes).

In place of `show`, you can also define the `hide` property. The possible values are same as in `show` property with the only difference that the icon is not displayed for the specified mode.

The following example shows XML Documentation solution version number when the user clicks on the Show Version icon in the toolbar.

Add the following code to a JavaScript file:

```
$(document).ready(setTimeout(function() {
    fmxml.commandHandler.subscribe({
        key: 'user.alert',
        next: function() {
            alert("XML Documentation solution version x.x")
        }
    })
}, 1000))
```

Add the feature in the ui_config.json file as:

```
"type": "button",
"icon": "alert","variant": "quiet","title": "About XML Documentation solution","show": “true”,"on-click": "user.alert"
```

6) Create a *clientlib* folder and add your JavaScript into this folder.
7) Update the categories property of the *clientlib* folder by assigning it the value of *apps.fmdita.xml_editor.page_overrides*.
8) Save the *ui_config.json* file and reload the Web Editor.

Remove a feature from the toolbar

At times you might not want to give all features currently available in the Web Editor, in that case you can remove the unwanted feature from the Web Editor’s toolbar.

Perform the following steps to remove any unwanted feature from the toolbar:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:.
   /ibs/fmdita/clientlibs/clientlibs/xmleditor/ui_config.json
3) Create a copy of the default configuration file at the following location:
   /apps/fmdita/xmleditor/ui_config.json
4) Navigate to and open the `ui_config.json` file in the `apps` node for editing.
   The `ui_config.json` file has three sections:

toolbars

This section contains the definition of all features available in the editor's toolbar such as Insert/Remove Numbered List, (file) Close, Save, Comments and more.

shortcuts

This section contains the definition of keyboard shortcuts assigned to a particular feature in the editor.

templates

This section contains the predefined structure of DITA elements that you can use in your document. By default, the templates section contains template definitions for a paragraph, simple table, table, and body elements. You can create a template definition for any element by adding a valid XML structure for the desired element.

5) From the toolbars section, remove the entry of the feature that you do not want to expose to your users.
6) Save the *ui_config.json* file and reload the Web Editor.

Customize WYSIWYG view

In addition to customizing the functionality exposed by the Web Editor, you can also customize the look-and-feel of your document when it is previewed or opened for editing in the Web Editor.

Perform the following steps to customize the rendition of DITA document while previewing or editing it in the Web Editor:

1) Log into AEM and open the CRXDE Lite mode.
2) Create a new node, say `web-editor-custom-css`, of type `cq:ClientLibraryFolder`. This node is used to store the customization files.
3) Add the following property to your node (`web-editor-custom-css`):

Property name	Type	Value
categories	String	Specify the value as: `apps.fmdita.xml_editor.content_overrides`

4) Create a new file named `css.txt`.

   NOTE: *The default CSS file for the Web Editor is placed in `/libs/fmdita/ clientlibs/previewtopic/css/previewtopic.css` folder and the override file is placed in `/libs/fmdita/clientlibs/clientlibs/ xmleditor/default-content/overrides.css` folder.*
5) Create a CSS file, say `overrides.css`, which will store the customized element definitions.

   If you want to customize an element's definition to be available only in the Web Editor, then it should be wrapped under the following code:

   ```
   body.cke_editable {...}
   ```

   Similarly, if you want to customize an element's definition only for previewing a document, then add your customizations in the following code:

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

   You can customize any element's definition by using its name and adding custom display attributes to it. For example, to customize a list element use `.li` and add display attributes to it.
6) Enter the name of the file you created in Step 5 (`overrides.css`) in `css.txt`.

7) In the `overrides.css` file, provide your custom settings to display DITA content. In the following example, the topic's title has been customized to display in red color and text marked as bold in a paragraph appears in green color.

```
.title{
color:red}

.p .b{
color:green}
```

8) Save all changes and open any topic in Preview or edit mode to see the changes.

Configure file auto-save in the Web Editor

One of the most common features in the browser-based editor it the ability to save data after a specific period of time. The XML Documentation solution's Web Editor also supports auto-saving of topic and map files at the specified time interval. When this feature is triggered, the working copy of the topic or map is saved. A new version of the topic or map is not created. To create a new version, you have to click the Save Revision icon in the Web Editor's toolbar.

The auto-save feature is not enabled by default and you need to enable this from the configMgr. Perform the following steps to enable the auto-save feature in the Web Editor:

1) Open the Adobe Experience Manager Web Console Configuration page.
   The default URL to access the configuration page is:
   `http://<server name>:<port>/system/console/configMgr`
2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.
3) In the *XmlEditorConfig* settings, select the **Auto Save** option.
4) In the **Auto Save Interval** field, specify the time interval in seconds to trigger the auto-save feature.
5) Click **Save**.

Auto-generate element IDs

XML Documentation solution generates a document ID for any new document that you create. For example, when you create a DITA map, an ID like `map.ditamap_random_digits` is assigned to the map's ID. You can also define elements on which an ID is automatically generated and assigned.

XML Documentation solution provides easy configuration settings wherein you need to define the elements on which an ID is auto-generated and a pattern for the ID. By default, some elements like `section`, `table`, `ul`, `ol`, are configured for auto-generation of ID. You can add other elements to this list so that whenever these elements are inserted in a document, XML Documentation solution generates and assigns an ID based on the given pattern

Perform the following steps to configure elements to have auto-generated ID:

1) Open the Adobe Experience Manager Web Console Configuration page.

   The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.

3) In the *XmlEditorConfig* settings, specify one or more elements in the **Auto Generate IDs for Element Tags** field.

NOTE: Element tags are the DITA element names such as `body`, `title`, `codeblock`, and so on. To specify multiple elements, separate element names with a comma.

4) In the**Pattern for Generating IDs** field, specify a pattern to generate an ID.

   The default value for this field is set to `${elementName}_${id}`. The `${elementName}` value is replaced with the name of the element. The `${id}` variable generates sequential number for the element. For example, if you assign the paragraph element to have auto-generated IDs, then the first paragraph in the topic or document will get an ID like `p_1`, the next paragraph will get `p_2`, and so on. However, in a different document, the ID generation process restarts. This means that in a different document, IDs like `p_1` and `p_2` can be assigned to paragraph elements.

   If your document already contains IDs in the specified pattern, then the auto-generation process does not assign those IDs to new elements.

5) Click **Save**.

Configure allowed special characters

The Web Editor allows you to insert some special characters out-of-the-box. However, you can customize the list of special characters that your authors can insert in their documents. If you customize the special characters list, then it overwrites the default set of special characters. Only those special characters that you add in your configuration are made available to the authors.

Perform the following steps to overwrite the default list of special characters:

1) Log into AEM and open the CRXDE Lite mode.

2) create `symbols.json` file at the following location:.

   `/apps/fmdita/xmleditor/`

3) Add the special character definition in the `symbols.json` file as:

```
{"symbols": [{"label": "Arrows",
   "items": [
   {   "name": "←",   "title": "Left Arrow"   }
   ,
   {   "name": "↑",   "title": "Up Arrow"      }
   ]
   }   ]   }
```

The structure of the `symbols.json` file is explained below:

- `"label": "Arrows"`: This specifies the category for the special characters. In the snippet, a category with the name `"Arrows"` is defined.

- `"items"`: This defines the collection of special characters in the category.
- `"name": "←", "title": "Left Arrow"`: This is the definition of the special character. It starts off with the `"name"` label, which must not be changed. The name is followed by the special character. The `"title"` is the name or title of the special character that appears as the tooltip for that special character.

  You can define multiple definitions of special characters within a category.

Configure filters for file browse dialog

While working in the Web Editor, you need to use the file browse dialog to insert elements like image, reference, or key reference. The default file browse dialog does not offer any file filtering option. You can add your own filters that would allow you to access the required files easily and quickly.

Perform the following steps to add your custom file filtering options to the file browse dialog:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default configuration file available at the following location:
   /ibs/fmdita/clientlibs/clientlibs/xmleditor/ui_config.json
3) Create a copy of the default configuration file at the following location:
   /apps/fmdita/xmleditor/ui_config.json
4) Navigate to and open the `ui_config.json` file in the `apps` node for editing.
5) In the `ui_config.json` file, add the definition of the filters you want to add.

   The following code snippet shows how to add two filtering options — DITA Files and Image Files.

```
"browseFilters": [
    {
      "title": "DITA Files",
      "property": "jcr:content/metadata/dita_class",
      "operation": "exists"
    },
    {
      "title": "Image Files",
      "property": "jcr:content/metadata/dc:format",
      "value": [
        "image/jpeg",
        "image/gif",
        "image/png"
      ]
    }
]
```

   In the above code snippet, the first filter is for DITA Files. The filter definition takes the following parameters:

title

The display name of the filter. This title appears as the filtering option in the file browse dialog.

property

The property to match in the file's metadata. For example, to allow only those files that have the `dita_class` metadata in their property, the property filter takes "`jcr:content/metadata/dita_class`" as its value.

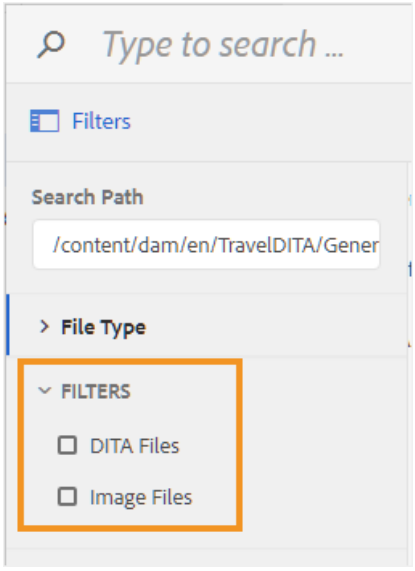
operation

Specify "`exists`" to match for existence of the value specified in the property parameter.

The second filter is for Image Files. The parameters are similar to the first filter except the `value` parameter. The `value` parameter takes an array of image types as its value. All file types specified in the value parameter are searched for and shown in the file browse dialog, all other file types are ignored.

6) Save the *ui_config.json* file and reload the Web Editor.

When you launch the file browse dialog, the filter options configured in the ui_config.json file are shown.



Configure global or folder-level profiles

In an enterprise, different groups or products may use different authoring templates, output templates, and conditional attribute profiles (or subject schemes). Configuring these only at an enterprise (or global) level can make authors experience difficult, as they will see templates or profiles that are not relevant to them.

XML Documentation solution allows you to configure authoring (topic or map) templates, output templates, and conditional attribute at an enterprise (global) level as well as at a folder level. This way, you can segregate the configurations for different departments or products in your enterprise.

Also, you can delegate the folder-specific configurations to a department or product administrators to decentralize the administration.

Using the Folder Profiles tile in the XML Documentation settings, you can configure setting under the following tabs:

- **General**: The general tab is available when you are configuring project/product/folder-level settings. You can configure settings such as the folder paths on which the settings will be applicable and users who will have administrative rights to create or update configurations.
- **Conditional Attributes**: Use this tab to configure conditional attributes at global or enterprise level. A conditional attribute is a combination of the attribute name and value, and you can also define a label for it. The conditional attributes that you define here are available to all users across projects.
- **Authoring Template**: Use this tab to configure the templates that your authors will use to create DITA content. The following topic templates are available out-of-the-box:
  - Glossary
  - Reference
  - Topic
  - Concept
  - Task
  - Troubleshooting
  - Blank
  - DITAVAL

  NOTE: You can use any of the existing templates as a base to create new templates. The Blank DITA template does not contain any structure or elements like the other templates. You can use any of the OOTB DITA template as the base, make modification to it, and save it with a different name. After making the required changes, add the updated template to the global or folder-level authoring templates configuration, then it becomes available for authoring.

  Along with topic templates, you can also define the map templates that will be made available to authors. The following map templates are available out-of-the-box:
  - Map
  - Bookmap
- **Output Preset**: Similar to Authoring Templates, there are five pre-configured output presets:

- AEM Site
- PDF
- HTML5
- EPUB
- Custom

Publishers can use these out-of-the-box output presets to publish content. These presets can be configured by an administrator of the global or folder-level profile. Once configured, the publishing presets become available to the publishers for newly created DITA maps. You can also apply publishing presets to existing DITA maps, see Apply preset changes for more details.

Similar to the global profile, you can create folder-level profiles. In a folder-level profile, you can define the folders on which the settings will be applicable. These settings include the conditional attributes, authoring templates, and output presets. The conditional presets and authoring templates are then made available to authors who work in the configured folders. Similarly, publishers will have access to the configured output presets defined within the configured folders.

A folder-level profile overrides the settings configured in the global profile. In other words, if a folder has a folder-level profile, then it will show the authoring templates and output templates configured in its corresponding folder profile. It will not show the templates configured in the global profile. However, this does not apply to the conditional attributes. In case of conditional attributes, the conditional attributes are merged at global and folder levels.

The following sections walk you through the process of configuring global profile and folder-level profiles.

Configure global profile

Perform the following steps to configure the global profile:

1) Log into Adobe Experience Manager as an administrator.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click the **Folder Profiles**.
   For the first time the Folder Profiles page is shown with only the Global Profile tile.
4) Click on the **Global Profile** tile.
5) To configure **Conditional Attributes**, see *Configure conditional attributes for global or folder-level profiles*.
6) To configure **Authoring Template**, see *Configure authoring templates*.
7) To configure **Output Presets**, see *Configure output presets*.
8) After making all required updates, save and close the **Global Profile**.

Create and configure a folder-level profile

Perform the following steps to configure a folder-level profile:

1) Log into Adobe Experience Manager as an administrator.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click on the **Folder Profiles** tile.
   For the first time, the Folder Profiles page is shown with the default Global Profile tile only.
4) Click **Create**.
5) Enter the following details in the **Create Folder Profile** dialog:
   – Name of the folder profile.
   – Path of the folder where the profile will be applicable.

     NOTE: You cannot apply multiple folder profiles on a folder. Ensure that the folder you are selecting here does not have any other profile applied to it. In case of a parent-child folders having their own specific profiles, the child folder will use the configurations from its own profile. The configurations from the parent folder do not override the configurations of a child folder.
6) Click **Create**.
   A new tile with the name of the folder profile is created in the Folder Profiles page
7) Click on the folder profile tile to edit.
   A General tab with the folder profile's name and configured folder information is shown.
8) Click **Edit** to add multiple folders and users who will have administrative access to modify the folder profile.

   NOTE: Users that you add here will have the administrative rights to update the conditional attributes, authoring template, and output presets configured for this folder profile.
9) To add a folder, click the Browse icon in the Folder Path and navigate to and select a folder, and click Add to add the folder to this profile.

   NOTE: Ensure that the folder that you choose here does not have any other folder-level profile associated with it.
10) To add a user, select a user from the **Admin Users** drop-down and click **Add**.

    NOTE: You can add multiple users to the folder profile from the drop-down list. You can also remove an existing admin user from the list by clicking the delete icon next to the user ID.
11) After adding all required folders and users to the folder profile, click **Save**.

Now you are ready to configure the conditional attributes, authoring templates, and output presets.

IMPORTANT: When you create a folder profile, by default it does not contain any authoring templates. You must add the required authoring templates in the folder profile to make them available to your authors.

Configure conditional attributes for global or folder-level profiles

Perform the following steps to configure global or folder-level conditional attributes:

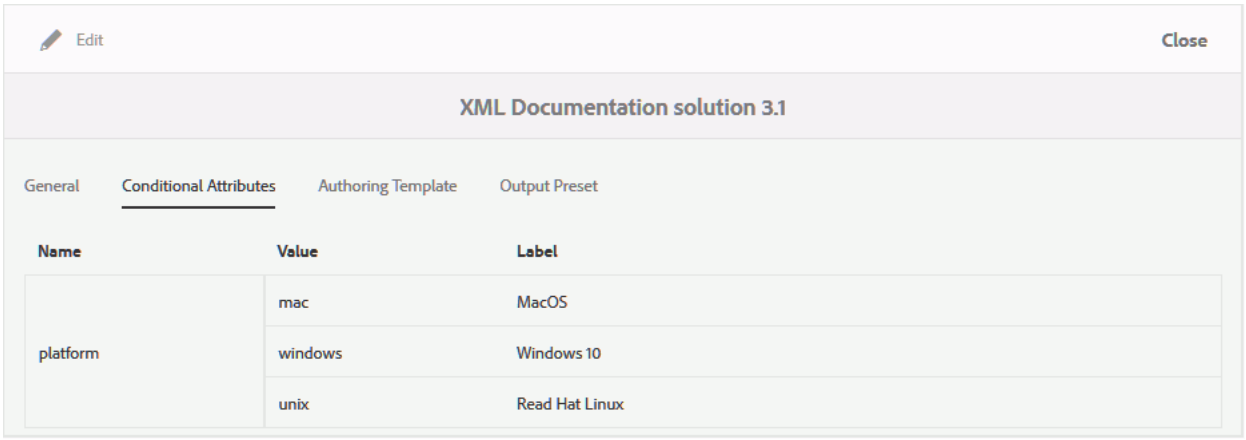
1) Log into Adobe Experience Manager as an administrator or the user having administrative rights on a folder-level profile.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.

   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click on the **Folder Profiles** tile.
4) Click on the profile tile that you want to configure.

   NOTE: You can choose to configure conditional attributes in the Global Profile or a folder-level profile.
5) On the profile page, click on the **Conditional Attributes** tab.
6) Click **Edit**.
7) Click **Add**.
8) Enter the **Name**, **Value**, and a **Label** for the conditional attribute.

   You can save a profile with only the attribute name. However, an attribute can only be used when it has a value specified to it. If you specify both - value and label for an attribute, the Web Editor shows the label of the conditional attribute. Also, the label is shown to the publishing administrator at the time of creating conditional preset.

   The following screenshot shows the definition for the `platform` attribute with possible values and labels.



9) If you want to add more values for the same attribute, click the **+** icon and enter additional value and label.
10) If you want to add more attributes, click **Add**.
11) Click **Save**.

NOTE: If you are using custom attributes (other than the default attributes provided in DITA specification) to conditionalize content, then you must explicitly define those in the `/libs/fmdita/config/condAttrList.xml` file. For making any customizations, create an

overlay of the `config` node in the `apps` folder and update the `condAttrList.xml` file in the `apps` folder.

Configure authoring templates

XML Documentation solution comes with 7 out-of-the-box authoring templates and 2 DITA map templates. You can choose to have only a few templates available to your authors. In case you use a custom template, the same can be configured and made available for authoring. You use the Authoring Template tab in the Folder Profiles configuration to add or remove topic or map templates from global or folder-level profiles.

Even before configuring the topic or map templates at global or folder-level, you can also define a location to store your custom authoring templates. To configure a custom location to store authoring templates, see *Configure custom DITA template folder path*.

Perform the following steps to add the topic or map templates into a folder profile:

1) Log into Adobe Experience Manager as an administrator or the user having administrative rights on a folder-level profile.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.

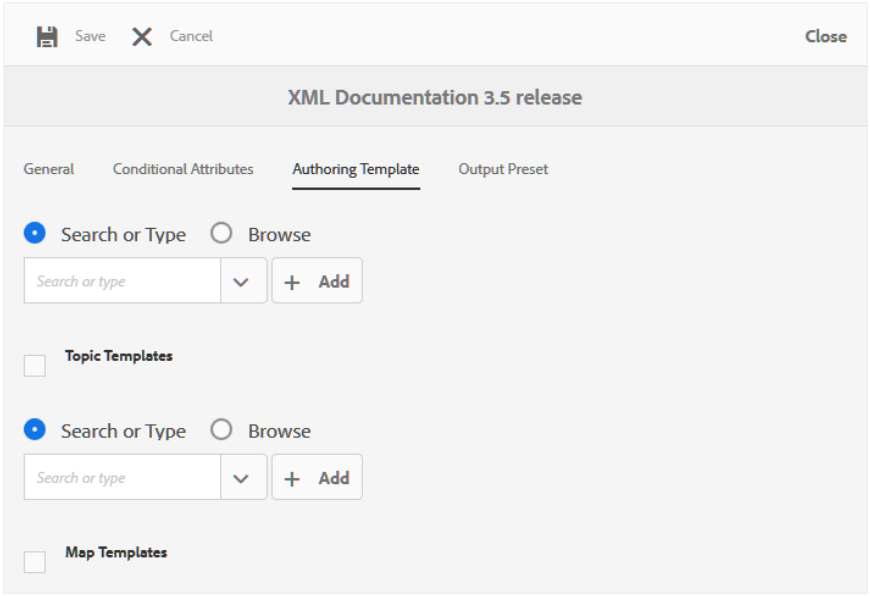
   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click on the **Folder Profiles** tile.
4) Click on the profile tile that you want to configure.

   NOTE: You can choose to configure authoring template in the Global Profile or a folder-level profile.
5) On the profile page, click on the **Authoring Template** tab.
6) Click **Edit**.

   You get the options to add Topic and Map templates by searching from the default location or browsing for it.

   ***NOTE:** By default, all authoring templates are stored in the `/content/dam/dita-templates` folder. The `dita-templates` folder contains `topics` and `maps` sub-folders to store the topic and map templates. You can add your custom templates (`.dita`, `.xml`, or `.ditamap` files) in the default template folders. Once you add your template in the default folder, you will be able to add*

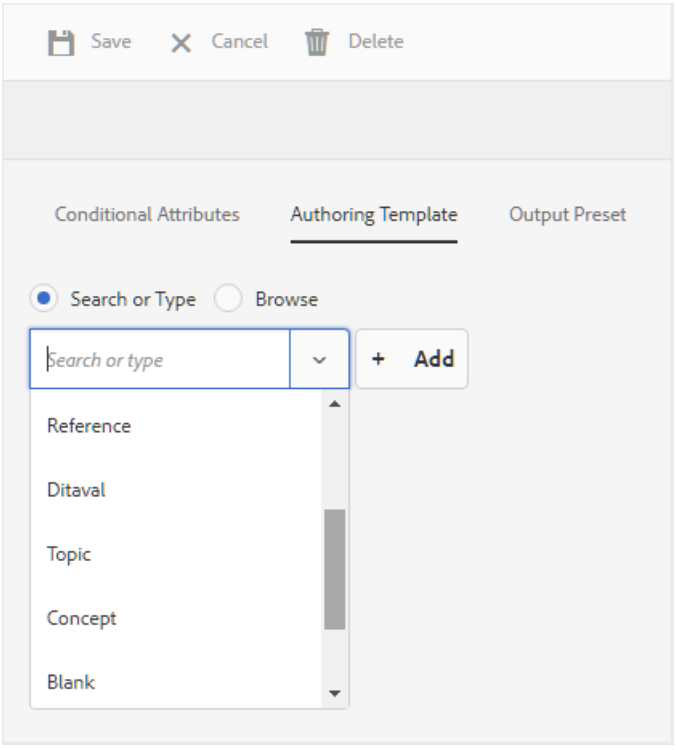
them in the global or folder profile. For more information about creating custom templates using the Web Editor, see Create custom authoring template.



7) Add the required topic and map templates to your profile.

To add a template, do one of the following:

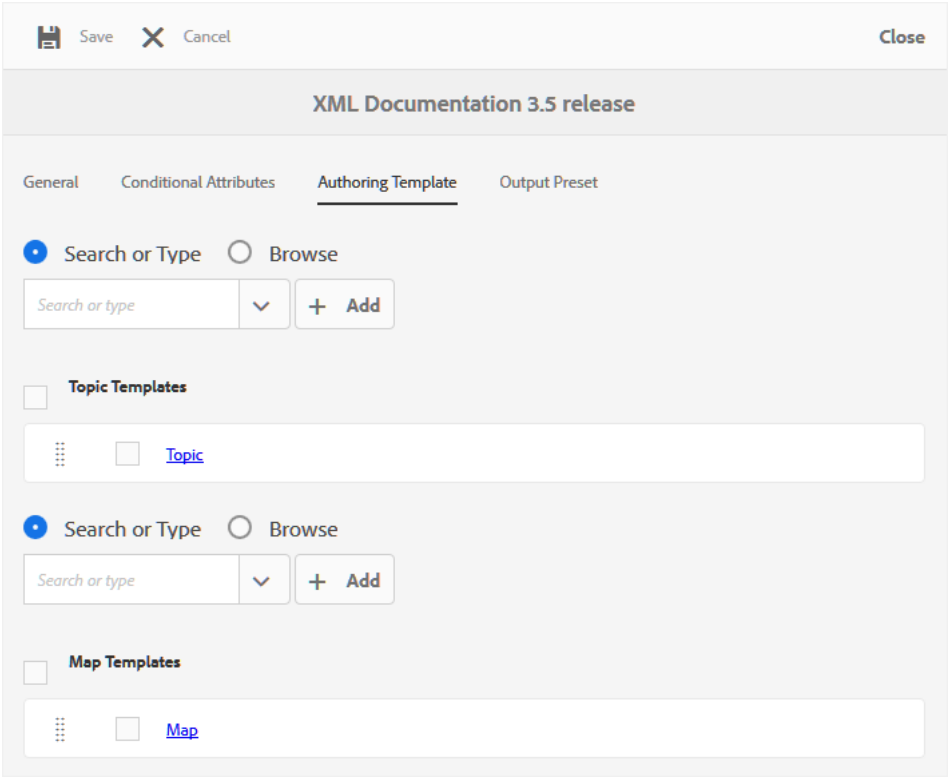
– Choose **Search or Type** and enter or select the name of a template from the drop-down list. The drop-down list consists of all default templates and any new template that you have created.



– Click **Browse** and select a template from DAM.

8) Click **Add**.

The selected templates are added to the template list.



NOTE: In AEM 6.4 and 6.3 you can change the order of templates by dragging and dropping them at the desired position in the list. The position of templates controls the order in which they show in the Blueprint page in the topic or map creation workflow.

9) Click **Save**.

In case you have configured the templates on a folder-level profile, the configured templates get associated with the configured folder. All projects created under the configured folder will have access to only those templates that are configured under the folder-level profile.

Create custom authoring template

XML Documentation solution provides an easy way of creating authoring templates. As a system administrator, you can use the Web Editor to create authoring template from scratch. You can then add the new template in the global profile or assign it to a specific folder using the folder-specific profile.

Perform the following steps to create a custom authoring template:

1) Log into Adobe Experience Manager as an administrator.
2) In the Assets UI, navigate to the folder configured to store the template files. By default, all topic templates are stored in the `/content/dam/dita-templates/topics` folder.

   NOTE: To configure a custom location to store topic or map templates, see *Configure custom DITA template folder path*
3) Click **Create** > **DITA Template**.
4) On the Blueprint page, select the type of the DITA topic template that you want to create.

NOTE: You can use the Blank template to start from scratch. The Blank template does not have any structure or elements in it.

5) Click **Next**.

6) On the new template Properties page, enter a **Title**, **Name**, and **Description** for the template.

   NOTE: *The name is automatically suggested based on the Title of your template. If you want to manually specify the name, then ensure that the Name does not contain any spaces, apostrophe, or braces and ends with* `.dita`*.*

7) *(Optional)* Click the **Add a Thumbnail** button to browser for and select a thumbnail to associate with your template.

8) Click **Create**.

   The Topic Created message appears.

   You can choose to open the template for editing in the Web Editor, or save the template file in the template store location. Once the template is created, you can use the Web Editor to customize the template as per your authoring needs. Once a template is in place, ensure that you associate it either with a global or folder-level profile.

Configure output presets

In a typical enterprise setup, different output templates could be used for different products or user guides. Also, there could be some common output generation processes that should be used by all publishers and a set of specific output generation processes for a specific group of publishers or projects.

XML Documentation solution allows the administrator to create output presets with specific settings that can then be used by all or a specific set of publishers to generate output. For example, the administrator can create one output preset to generate user guides that is common across all publishers. And, another one to create the programming user manuals that is specific to a set of publishers. Both of these presets can be configured to use different output templates. In this example, the common publishing preset for generating the user guide can be configured at the global level. And, the output preset for generating programming user manual can be configured at a folder-level.

Once the default output presets have been created in the system, all DITA maps created after that will use the default presets to generate output. However, all existing DITA maps would continue to use the output presets that were earlier configured with them. If you want to apply the new output preset on all existing DITA maps, then you need to run the Apply preset changes workflow.

In addition to the presets configured at the global or enterprise level, publisher would still have the rights to create more output presets. However, those presets are tied to the DITA map for which they are created. For more details about creating regular output presets for a DITA map, see *Create, edit, duplicate, or remove an output preset* in XML Documentation for Adobe Experience Manager User Guide.

Perform the following steps to configure global or folder-specific output presets:

1) Log into Adobe Experience Manager as an administrator or the user having administrative rights on a folder-specific profile.

2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.

   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*

3) Select **XML Documentation** from the list of tools and click on the **Folder Profiles** tile.
4) Click on the profile tile that you want to configure.

   NOTE: You can choose to configure output presets in the Global Profile or a folder-specific profile.
5) On the profile page. click on the **Output Presets** tab.

   A list of out-of-the-box output presets is displayed, which includes AEM Site, PDF, HTML5, EPUB, and CUSTOM.
6) Do one of the following to create or edit an output preset:
   - Click **Create** to create a new output preset from scratch.
   - Click Duplicate to create a copy of the selected output preset. You can make changes to the duplicate preset and save it.
   - Click **Edit** to open the selected preset's configuration for editing.

     For information about output preset settings, see Understanding the output presets in XML Documentation for Adobe Experience Manager User Guide.
7) Click **Save** to save the preset settings.

All DITA maps created or uploaded after this will have the new or updated output preset.

Apply preset changes

A new output preset created at the global level is made available to all new DITA maps that you create going forward. Similarly, if a new output preset is created at a folder-level, then that preset is made available to all maps that will be created in the configured folder. By default, a new output preset is not made available to any existing DITA map.

If you have updated and existing output preset, or you want to make a new output preset available to existing DITA maps, then perform the following steps:

1) Log into Adobe Experience Manager as an administrator or the user having administrative rights on a folder-specific profile.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the Adobe Experience Manager link at the top and choose **Tools**.

   *(In AEM 6.1) Click on the Experience Manager link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click on the **Folder Profiles** tile.
4) Click on the profile tile that you want to configure.

   NOTE: You can choose to configure output presets in the Global Profile or a folder-specific profile.
5) On the profile page. click on the **Output Presets** tab.

   A list of out-of-the-box output presets is displayed, which includes AEM Site, PDF, HTML5, EPUB, and CUSTOM.
6) Select the output preset that you want to apply to existing DITA maps.
7) Click **Apply Preset Changes** in the main toolbar.
8) In the *Apply Preset Changes* dialog, you can choose from:
   - **Selecting Overwrite Existing Preset option**: If you select this option, then any updates that you made in the existing output presets will overwrite settings in all existing DITA maps where

that preset is used. However, doing so will result in loss of any existing conditional preset and baseline information associated with the map.

– **Not selecting Overwrite Existing Preset option**: If you do not select this option, then any updates that you made in the existing output presets will not impact the existing DITA maps. Only the newly added presets are added to the existing DITA maps. Note that newly created DITA map get both—the updated output presets and the newly added presets.

9) Click **OK** to apply changes from the selected output presets on all existing DITA maps.

Version management

Versioning is an important aspect of any content management system. It allows you to create a snapshot of your digital asset at a specific point in time. With a version of a digital asset in place, you can restore the required version of the asset and update it. Typically, for creating a version of any asset, you would check out and check in the required asset.

As an administrator, you can enforce rules that will restrict users from editing a file without checking it out. Similarly, you can ensure that all checked out files are checked in back to avoid any data loss.

In a multi-use environment, it is also important to ensure that users do not delete files from the system. This requirement is more critical for files that are checked out by other users. To prevent users from accidentally deleting checked out files from the system, XML Documentation solution provides a configuration that you can use.

Configure settings to allow editing of checked out files

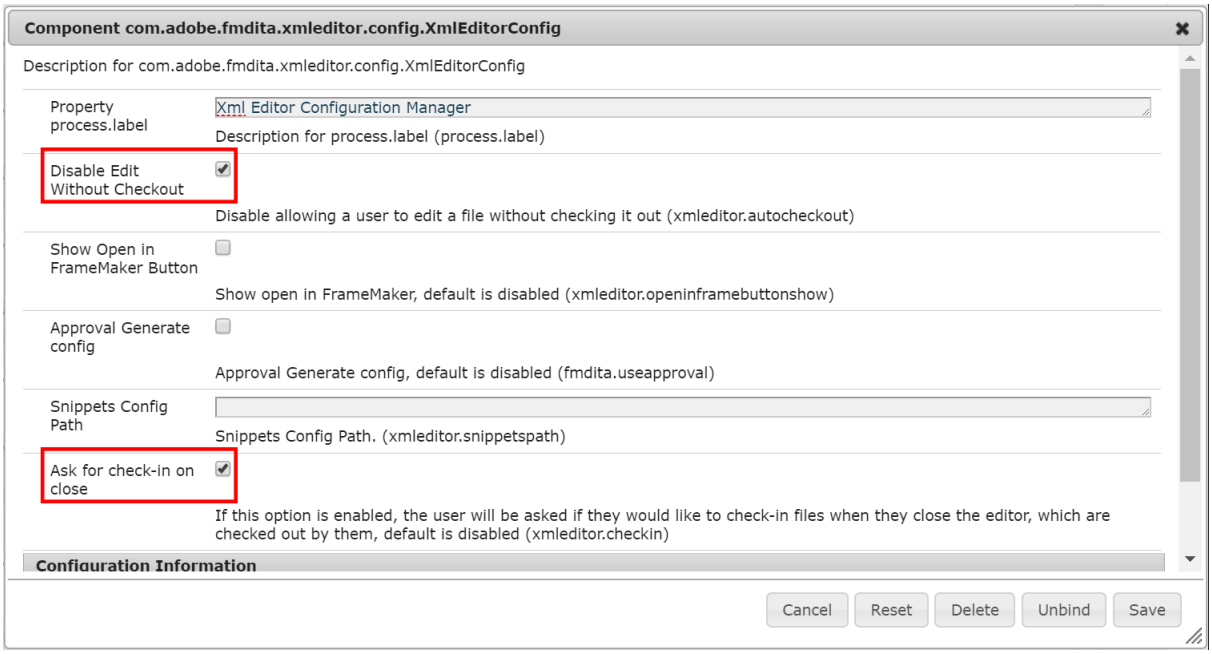
The XML Documentation solution's Web Editor allows you to create and update DITA topics. You can configure the Web Editor to allow editing of only those documents that have been checked out from the repository. This ensure that no other writer accidentally overwrites a topic that is opened for editing by another writer. Once a topic is opened for editing, an author can check-in the file at the time of closing the file.

Another important rule is to ensure that files that have been checked out are checked back into the system. This prevents users from accidentally closing the files without checking them back in.

Perform the following steps to enable these features:

1) Open the Adobe Experience Manager Web Console Configuration page.
   The default URL to access the configuration page is:
   `http://<server name>:<port>/system/console/configMgr`
2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.
3) Select the **Disable Edit Without Checkout** option.

With this option, users will not see the Edit option in the toolbar until they check out a file.



4) Select the **Ask for Check-in on Close** option to show a warning message whenever a checked out file is closed without saving or checking it back into the repository.
5) Click **Save**.

NOTE: Irrespective of whether you turn on or off this feature, the file Check Out and Check In options are always available in a topic preview.

Prevent deletion of checked out files

To prevent users from accidentally deleting files that have been checked out by them or some other user, perform the following steps:

1) Open the Adobe Experience Manager Web Console Configuration page.
   The default URL to access the configuration page is:
   `http://<server name>:<port>/system/console/configMgr`
2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.
3) Select the **Prevent Deletion of Checked Out Content** option.
   With this option selected, users will not be able to delete checked out files.
4) Click **Save**.

Integrate desktop-based XML editors

There are a lot of XML editors available on the market, and you could be using one already. Adobe FrameMaker is one of the most powerful XML editors, which comes with AEM connector. Using the AEM connector in FrameMaker, you can easily connect with AEM repository, check-out and check-in files, and edit files directly in FrameMaker. You can also configure XML Documentation solution to launch FrameMaker from the Web Editor. Once you have the file opened in FrameMaker, you can edit and check the file back into AEM repository.

Enable file editing in FrameMaker from the Web Editor

You can use FrameMaker or any other DITA editor to create and update DITA content. However, if your organization uses FrameMaker as DITA editor, then you can give your users an option to open DITA documents directly in FrameMaker from AEM.

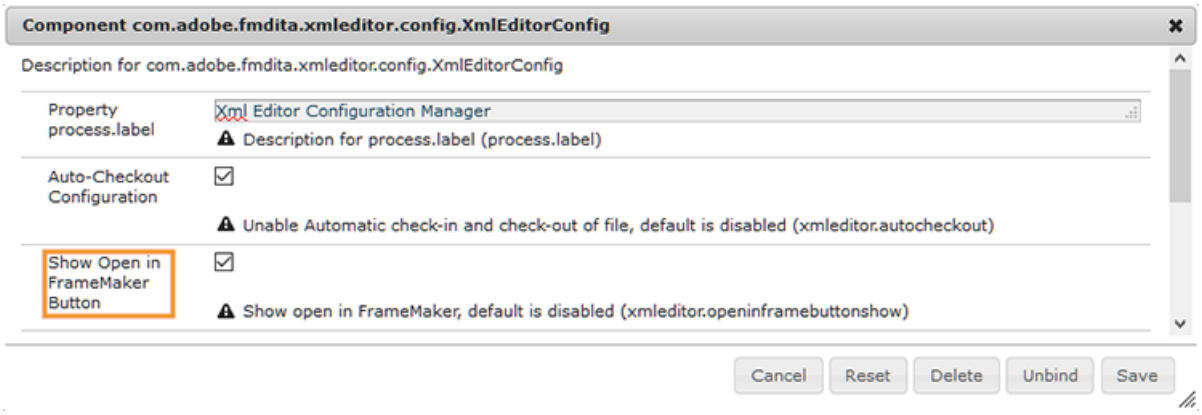
By default, your users do not see the **Open in FrameMaker** button on the AEM toolbar. Perform the following steps to add this button on the AEM toolbar:

1) Open the Adobe Experience Manager Web Console Configuration page.

   The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.



3) Select the **Show Open in FrameMaker Button** option.

4) Click **Save**.

When you enable the **Show Open in FrameMaker Button** option, then the **Open in FrameMaker** button is shown on selecting any DITA file in the AEM repository. When this option is *not enabled*, the **Open in FrameMaker** button is shown only when you select a `.fm` or a `.book` file in the repository.

Configure output generation settings

XML Documentation solution comes with a lot of configuration options for you to customize the output generation process. This topic covers all configurations and customizations that would help you set up your output generation process.

Configure FrameMaker Publishing Server

You can use FrameMaker Publishing Server (FMPS) to generate output for your DITA content. Configuring FMPS will allow you to generate output in multiple formats supported by FMPS.

NOTE: To generate output using FMPS, you need to have the FMPS server setup. For installation and configuration details, see the FrameMaker Publishing Server User Guide.

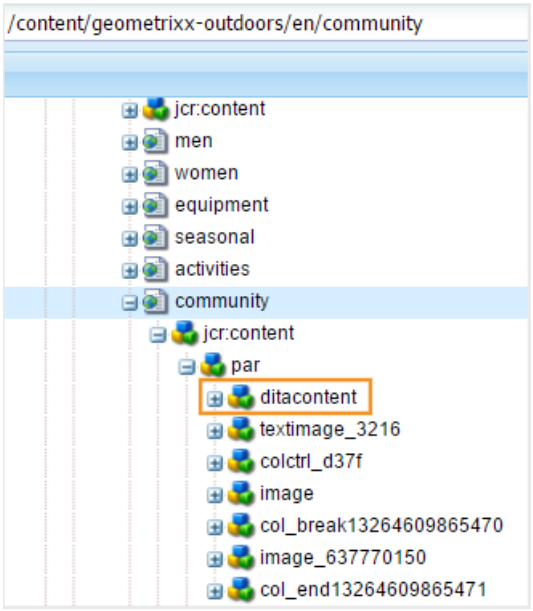
To configure your XML Documentation solution to use FMPS, update the following properties of the `com.adobe.fmdita.config.ConfigManager` bundle in the Web Console.

NOTE: Access `http://<server name>:<port>/system/console/configMgr` URL to open the Web Console.

Property	Description
FrameMaker Publishing Server Login Domain	Specify the domain name or the workgroup name on which the FrameMaker Publishing Server is hosted.
FrameMaker Publishing Server URL	Specify the URL of the FrameMaker Publishing Server. For example, `http://<fmps_ip>:<port>/fmserver/v1/`.
FrameMaker Publishing Server Username and Password	Specify the user name and password to access the FrameMaker Publishing Server.
FMPS Timeout	(*Optional*) Specify the time (in seconds) for which the XML Documentation solution waits for a response from the FrameMaker Publishing Server. If no response if received in the specified time, XML Documentation solution terminates the publishing task and the task is flagged as failed. Default value: 300 seconds (5 minutes)
External AEM URL	*(Optional)* The AEM URL where the FrameMaker Publishing Server will place the generated output files. For example, `http://<server-name>:4502`.
AEM Admin Username and Password	*(Optional)* The user name and password for an administrator of your AEM setup. This will be used by FrameMaker Publishing Server to communicate with AEM.

Configure blended publishing within an existing AEM Site

If you have an AEM Site that contains DITA content, you can configure your AEM Site output to publish DITA content to a predefined location within your site. For example, in the following screenshot of an AEM Site page, the `ditacontent` node is reserved to store DITA content:



The remaining nodes in the page are authored directly from the AEM Site editor. Configuring the publish setting to publish DITA content to a predefined location ensures that none of your existing non-DITA content gets modified by the XML Documentation solution publishing process.

You need to perform the following configurations on your existing site to allow publishing of DITA content to a predefined node:

- Configure your site's template properties
- Add nodes in your site to publish DITA content

Perform the following steps to configure your existing site's template properties:

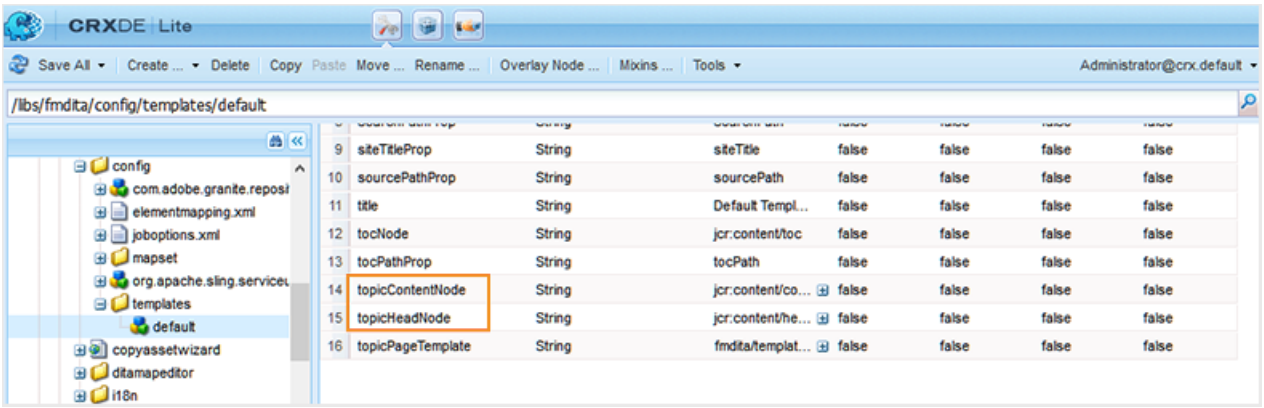
1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to your site's template configuration node. For example, the XML Documentation solution stores the default template configurations in the following node:

   `/libs/fmdita/config/templates/default`

   NOTE: *Do not make any customizations in the default configuration files available in the* `libs` *node. You must create an overlay of the* `libs` *node in the* `apps` *node and update the required files in the* `apps` *node only.*
3) Add the following properties:

Property name	Type	Value
`topicContentNode`	String	Specify the node name where you would like to publish the DITA content. For example, the default node where XML Documentation solution publishes DITA content is: `jcr:content/contentnode`
`topicHeadNode`	String	Specify the node name where you would like to store the metadata information of your DITA content. For example, the default node where XML Documentation solution stores metadata information is: `jcr:content/headnode`

The following screenshot shows the properties added in the default template node of XML Documentation solution:



Next time when you publish any DITA content using your site's template configurations, the content gets published into the nodes specified in the `topicContentNode` and `topicHeadNode` properties.

However, for existing sites, you must manually add the `topicContentNode` and `topicHeadNode` nodes.

Perform the following steps to add the required nodes to your existing site:

1) Log into AEM and open the CRXDE Lite mode.
2) Locate `jcr:content` within your site node.
3) Add `topicContentNode` and `topicHeadNode` nodes with the same name that you specified in the site's template configurations.

Customize AEM Site output design template

The XML Documentation solution supports creating outputs in following formats:

- AEM Site
- PDF
- HTML5
- EPUB
- Custom output through DITA-OT

For the AEM Site output, you can assign different design templates with different output tasks. These design templates can render the DITA content in different layouts. For example, you could specify different design templates for internal and external audiences.

You can also use customized DITA Open Toolkit (DITA-OT) plug-ins with the XML Documentation solution. You can upload these custom DITA-OT plug-ins to generate PDF output in a specific way.

TIP: See *AEM Site publishing* for best practices around creating AEM Site output.

Customize design template for generating output

The XML Documentation solution uses a set of predefined design templates to generate AEM Site output. You can customize the XML Documentation solution design templates to generate the output that conforms to your corporate branding. A design template is a collection of various styles (CSS), scripts (both server- and client-side), resources (images, logos, and other assets), and JCR nodes that tie all these resources together. A design template can be as simple as a single server-side script with just a couple of JCR nodes, or a complex combination of styles, resources, and JCR nodes. Design templates are used by XML Documentation solution's publishing subsystem while generating AEM Site output and they control the structure, look and feel of the generated output.

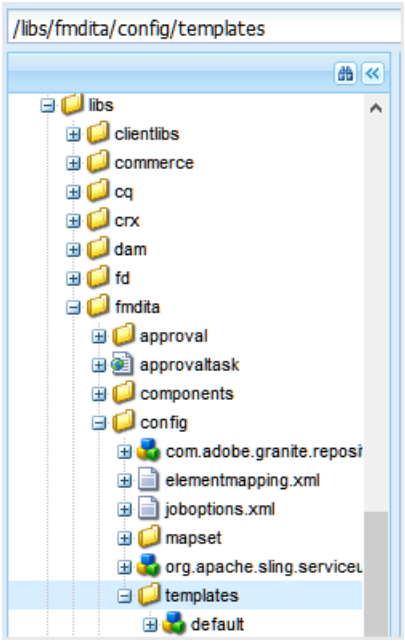
There is no restriction as to where the design template resources should be located on the server, but they are usually logically organized as per their function. For example, the default template has all its JavaScript and CSS files stored under `/etc/designs/fmdita/clientlibs/siteoutput/default` folder. Wherever these files are located, they are linked together by a collection of JCR nodes. Together, these JCR nodes and the files constitute the whole design template.

The default design template shipped with the XML Documentation solution allows you to customize the landing, topic, and search page components. You can make a copy of the default design and the corresponding reference templates and specify different components to generate the desired output.

Perform the following steps to specify your own design template to use for AEM Site output generation:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the default design template node. The location of the default design template node is:

/libs/fmdita/config/templates/



NOTE: Make a copy of the default design templates from the `libs` folder to the `apps` folder and make changes in the `apps` folder. You must also make changes in the templates referenced from the default template node. The referenced templates are placed under `/libs/fmdita/templates/default/cqtemplates` node. Make a copy of the referenced templates in the `apps` folder before making any changes.

3) Click the *default* component in the *templates* node to access its properties.

The XML Documentation solution design template properties are described in the following table.

Property	Description
`landingPageTemplate`, `searchPageTemplate`, `topicPageTemplate`, `shadowPageTemplate`	Specify the `cq:Template` node for these corresponding pages (landing, search, and topic). By default the `cq:Template` node for these pages can be found in `/libs/fmdita/templates/default/cqtemplates` node. This node defines the structure and properties of the landing, search, and topic pages. The `shadowPageTemplate` is used to optimize the chunked content. You need to set the value of this property to: `fmdita/templates/default/cqtemplates/shadowpage` **NOTE:** You must specify a value for the `topicPageTemplate`. The `landingPageTemplate` and `searchPageTemplate` are optional properties. If you do not want the search and landing pages to generate, do not specify these properties.
`title`	A descriptive name of your design template.

Property	Description
`topicContentNode`	Specify the location of the node that will contain the DITA content in a topic page. Path is relative to the topic page.
`topicHeadNode`	Specify the location of the node that will contain the head values (or metadata) derived from the DITA content. Path is relative to topic page.
`tocNode`	Specify the location of the node that will contain the TOC. Path is relative to the landing page or destination path.
`basePathProp`, `indexPathProp`, `pdfPathProp`, `pdfTypeProp`, `searchPathProp`, `siteTitleProp`, `sourcePathProp`, `tocPathProp`	Specify names for the corresponding properties to be set on the topic, landing, or search pages.

NOTE: After creating a custom design template node, you must update the Design option in the AEM Site output presets to use the custom design template node.

For more information, see Creating your First Adobe Experience Manager 6.3 website and The Basics of developing your own website on AEM.

Use metadata in publishing output through DITA-OT

XML Documentation solution provides a way to pass custom metadata while publishing output using DITA-OT. As an administrator and a Publisher you would need to perform the following tasks to configure and use custom metadata in the published output:

- As an administrator, add the required metadata in the system so that it is made available on the Properties page of the DITA map.
- As an administrator, add the custom metadata in the metadata list so that it shows up in the DITA map console.
- As a Publisher, configure and add the custom metadata with the DITA map and generate the required output.

To add the required metadata in the system, perform the following steps:

1) Log into Adobe Experience Manager as an administrator.
2) Click on the Adobe Experience Manager link at the top and choose **Tools**.
3) Select **Assets** from the list of tools.
4) Click on the **Metadata Schemas** tile.

   The Metadata Schema Forms page is displayed.

5) Select the **default** form from the list.

   NOTE: The properties displayed on the Properties page for a DITA map are taken from this form.

6) Click **Edit**.

7) Add the custom metadata that you want to use in your published outputs. For example, we will add `audience` metadata using the following steps:

   a) From the **Build Form** components list, drag-and-drop **Single Line Text** component onto the form.

   b) Select the new field to open the **Settings** of the field.

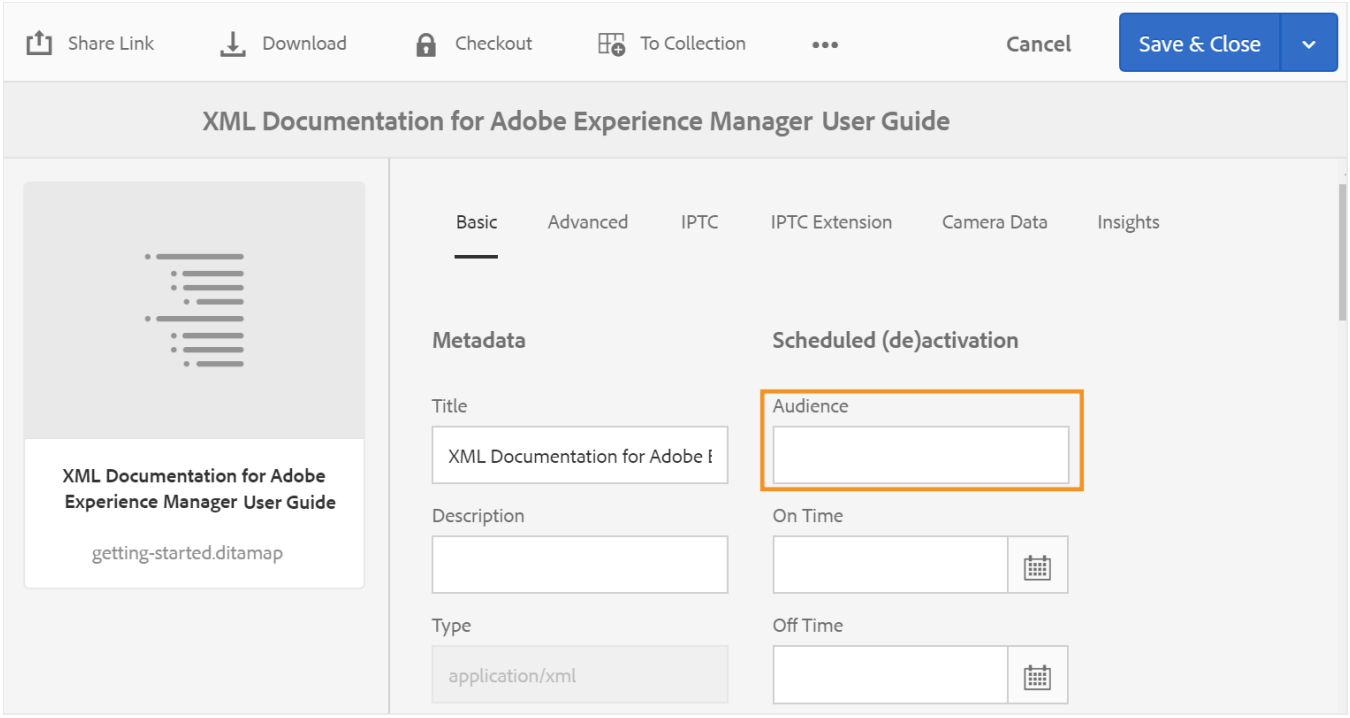
   c) In the **Field Label**, enter the metadata name— `Audience`.

   d) In the **Map to Property** setting, specify `./jcr:content/metadata/<name of the metadata>`. For our example, we will set it to `./jcr:content/metadata/audience`.

   Using these steps, add all required metadata parameters.

8) Click **Save**.

The new parameter now shows up on the *Properties* page for all DITA maps.



Next, you need to make the custom metadata available in the DITA map console. Perform the following steps to make the custom metadata available on the DITA map dash board:

1) Log into AEM and open the CRXDE Lite mode.

2) Access the `metadataList` file available at the following location:

   `/libs/fmdita/config/metadataList`

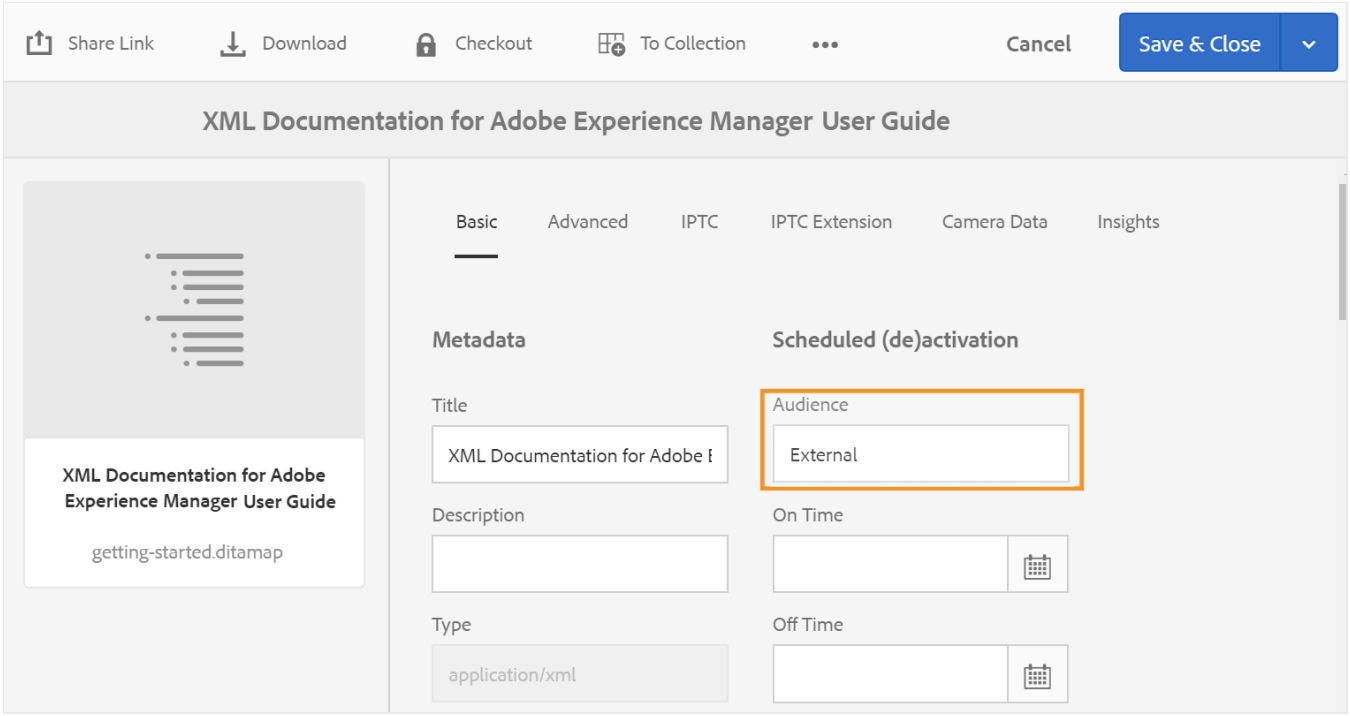
   ***NOTE:** The `metadataList` file contains a list of properties that are shown in the **Properties** drop-down list of a DITA map in the map dashboard. By default, there are four properties listed in this file— `docstate`, `dc:language`, `dc:description`, and `dc:title`.*

3) Add the custom metadata that you have added in the Metadata Schema Forms page. For our example, add `audience` parameter to the end of the default list.
4) Click **Save All**.

Now the custom metadata will show up in the DITA map console's **Properties** drop-down list.

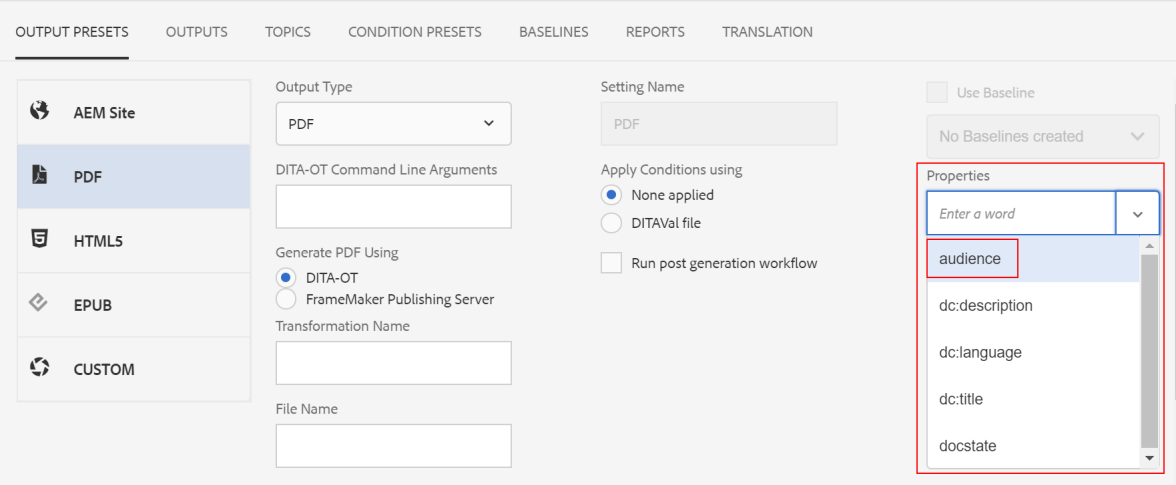
Lastly, as a Publisher, you need to include the custom metadata in the published output. To process the custom metadata while generating the output, perform the following steps:

1) In the Assets UI, navigate to the DITA map that you want to publish.
2) Select the DITA map file and open its properties page.
3) On the *Properties* page, specify the value for the custom metadata. For our example, we have specified a value of `External` for the `audience` parameter.



4) Click **Save & Close**.
5) Click on the DITA map file to open the DITA map console.
6) In the **Output Presets** tab, select the output preset that you want to use to generate the output.
7) Click **Edit**.

8) From the **Properties** drop-down list, select the properties that you want to pass on to the publishing process.



The selected properties/metadata are passed on to the publishing process and are made available in the final output.

Customize DITA element mapping with AEM components

DITA elements in the XML Documentation solution are mapped to their corresponding AEM components. XML Documentation solution uses this mapping in workflows such as publishing and review to convert DITA element to a corresponding AEM component. The mapping is defined in the `elementmapping.xml` file, which can be accessed from the CRXDE Lite mode. Access the following URL in the CRXDE Lite mode:

`/libs/fmdita/config/elementmapping.xml`

NOTE: Do not make any customizations in the default configuration files available in the `libs` node. You must create an overlay of the `libs` node in the `apps` node and update the required files in the `apps` node only.

You may use the predefined DITA element mappings, or you can map DITA elements to your custom AEM components. To use your custom AEM components, you need to understand the structure of the `elementmapping.xml` file.

elementmapping.xml structure

A high-level overview of the `elementmapping.xml` structure is explained below:

1) Every DITA element is first searched for a corresponding component mapping based on the element name. For example:

```
<ditaelement>
   <name>substeps</name>
   <class>- topic/ol task/substeps</class>
   <componentpath>dita/components/ditaolist</componentpath>
   <type>COMPOSITE</type>
```

```
   <target>para</target>
</ditaelement>
```

In the above example, all `substeps` DITA elements are rendered using the `dita/components/ditaolist` component.

2) If a DITA element does not find a match based on the name, then a match on the basis of the `class` is done. For example:

```
<ditaelement>
   <name>topic</name>
   <class>- topic/topic</class>
   <componentpath>fmdita/components/dita/topic</componentpath>
   <type>COMPOSITE</type>
   <target>para</target>
   <attributemap>
      <attribute from="id" to="id" />
   </attributemap>
</ditaelement>
```

In the above example, if there is no mapping defined for the `task` element, then the `task` element is mapped to the above component because `task` is inherited from the `topic` component.

3) When an element has a corresponding component mapping, then further processing of its child elements is determined by `type`. For example:

```
<ditaelement>
   <name>title</name>
   <class>- topic/title</class>
   <componentpath>foundation/components/title</componentpath>
   <type>STANDALONE</type>
   <target>para</target>
   <textprop>jcr:title</textprop>
</ditaelement>
```

`type` takes the following values:

– COMPOSITE: element to component *mapping continues for child elements* as well.

– STANDALONE: child elements of the current element are *not mapped further*.

In the above example, if the `<title>` element has any child elements, they will not be mapped to any other component. The component for `<title>` element is responsible for rendering all child elements inside the `<title>` element.

4) If there are multiple components mapped to a single DITA element, then the best match for the element is selected. To select the best match component, domain and structural specialization of DITA elements is considered.

If there are DITA elements with domain specialization and a component is mapped for domain specialization, then that component is given high priority.

Similarly, if there are DITA elements with structural specialization and a component is mapped for structural specialization, then that component is given high priority.

5) You can use `<attributemap>` in element mapping to map attribute values to the corresponding node properties.

6) `textprop` can be used for serializing the text content of a DITA element to a node property. In addition, it can be used multiple times in an element tag to serialize the text content at multiple locations in published hierarchy. You can also customize the location and name of the target property. For example:

```
<ditaelement>
  <name>title</name>
  <class>- topic/title</class>
  <componentpath>foundation/components/title</componentpath>
  <type>STANDALONE</type>
  <target>para</target>
  <textprop>jcr:title</textprop>
</ditaelement>
```

The above element mapping specifies that the text content of `<title>` element will be saved as value of a property named `jcr:title` on the output node.

7) `xmlprop` can be used for serializing the entire XML for a given element to a node property. The component can then read this node property and do custom rendering. For example:

```
<ditaelement>
  <name>svg-container</name>
  <class>+ topic/foreign svg-d/svg-container</class>
  <componentpath>fmdita/components/dita/svg</componentpath>
  <type>STANDALONE</type>
  <target>para</target>
  <xmlprop>data</xmlprop>
</ditaelement>
```

The above element mapping specifies that the entire XML markup for element `<svg-container>` will be saved as value of a property named `data` on the output node.

8) There is a special attribute mapping to handle path resolution in output generation process. For example:

```
<attributemap>
  <attribute from="href" to="fileReference" ispath="true" rel="source"
/>
  <attribute from="height" to="height" />
  <attribute from="width" to="width" />
</attributemap>
```

For the above `attributemap`, the `href` attribute in your DITA element will be mapped to a node property named `fileReference`. Now since `ispath` is set to `true`, the output generation process resolves this path and then sets it in `fileReference` node property.

How this resolution happens is determined on the basis of value of the `rel` attribute in attribute mapping.

– If `rel=source`, then value of `href` is resolved with respect to the DITA source file that is currently being processed. The value of `href` is resolved and placed in the value of `fileReference` property.

– If `rel=target`, then value of `href` is resolved with respect to the root publish location. The value of `href` is resolved and placed in the value of `fileReference` property.

If you do not want any pre-processing or resolution to happen on path attributes, then you need not specify the `ispath` attribute. The value is copied as is and the component can do the required resolution.

DITA element schema

Following is an example of the DITA element schema in `elementmapping.xml` file:

```
<ditaelement>
  <name>element_name</name>
  <class>element_class</class>
  <componentpath>fmdita/components/dita/component_name</componentpath>
  <type>COMPOSITE|STANDALONE</type>
  <attributeprop>propname_a</attributeprop>
  <textprop>propname_t</textprop>
  <xmlprop>propname_x</xmlprop>
  <xpath>xpath expression string</xpath>
  <target>head|para</target>
  <wrapelement>div</wrapelement>
  <wrapclass>class_name</wrapclass>
  <attributemap>
    <attribute from="attrname" to="propname" ispath="true|false"
     rel="source|target" />
  </attributemap>
  <skip>true|false</skip>
</ditaelement>
```

The following table describes the elements in the DITA element schema:

Element	Description
`<ditaelement>`	The top level node for each mapping element.
`<class>`	The class attribute of the target DITA element for which you are writing the component. For example, the class attribute for the DITA topic is: `- topic/topic`
`<componentpath>`	The CRXDE path of the mapped AEM component.
`<type>`	Possible values: • **COMPOSITE**: Process child elements as well • **STANDALONE**: Skips processing of child elements

Element	Description
<attributeprop>	Used for mapping serialized DITA attributes and values to AEM nodes as property. For example, if you have <note type="Caution"> element and the component that is mapped for this element has <attributeprop>attr_t</ attributeprop>, then the node's attribute and value is serialized to attr_t property of the corresponding AEM node (attr_t->type="caution").
<textprop>propname_t</textprop>	Save the getTextContent() output to property defined by propname_t. **NOTE:** This is an optimized property.
<xmlprop>propname_x </xmlprop>	Save serialized XML of this node to property defined by propname_x. **NOTE:** This is an optimized property.
<xpath>	If XPath element is provided in the element mapping, then along with element name and class the XPath condition should also be satisfied for the component mapping to be used.
<target>	Place for the DITA element in the crx repository at specified location. Possible values: • **head**: Under the head node • **text**: Under the paragraph node
<wrapelement>	The HTML element to wrap the contents within.
<wrapclass>	The element value to the property wrapclass.
<attributemap>	Container node containing one or more <attribute> nodes.
<attribute from="attrname" to="propname" ispath="true|false" rel="source|target" />	Maps the DITA attributes to AEM properties: • ***from***: DITA attribute name • **to**: AEM component property name • **ispath**: If the attribute is a path value (for example: *image*) • **rel**: If the path is the source or target **NOTE:** If attrname starts with %, then map attrname minus '%' to prop 'propname'.

Additional notes

- If you plan to override the default element mapping, it is recommended that you do not make the changes in the default elementmapping.xml file. You should create a new mapping XML file and place the file at another location, preferably inside custom apps folder that you create.
- In the elementmapping.xml file, there are many mapping entries referencing the fmdita/components/dita/wrapper component. Wrapper is a generic component that renders relatively simple DITA constructs using properties on their site node to generate relevant

HTML. It uses the `wrapelement` property to generate enclosing tags and delegates the child rendering to the corresponding components. This is useful in cases where you only want a container component. Instead of creating a new component that renders a specific container tag like ** ** or **<p>**, you can use the Wrapper component with the `wrapelement` and `wrapclass` properties to achieve the same effect.

- It is not recommended to save large amounts of text in String JCR properties. The optimized property type calculation in output generation ensures that large text content is not saved as string type. Instead, when content larger than a certain threshold needs to be saved, the type of the property is changed to binary. By default, this threshold is configured to `512` bytes, but can be changed in the Configuration Manager (*com.adobe.fmdita.config.ConfigManager*) by changing the **Save as Binary Threshold** setting.
- If you are planning to override some (and not all) of the element mappings, you do not have to replicate the entire `elementmapping.xml` file. You need to create a new XML mapping file and define only the elements that you are overriding.
- After you have created the XML file in the custom location, update the `Override Element Mapping` setting in the `com.adobe.fmdita.config.ConfigManager` bundle.

Customize DITA map console

XML Documentation solution gives you the flexibility of extending the capabilities of the DITA map console. For example, if you have a set of reports that are different from what is available in XML Documentation solution, you can add such reports to the map console. To customize the map console, you need to create an AEM Client Library (or ClientLib) that will contain the code to perform the functionality that you need.

NOTE: Direct modifications to page components is not recommended, as it will get overwritten by new releases of the product.

XML Documentation solution provides the `apps.fmdita.dashboard-extn` category for customizing map console. Whenever the map console is loaded, the functionality created under the `apps.fmdita.dashboard-extn` category gets executed and loaded.

NOTE: For more information about creating AEM Client Library, see Using Client-Side Libraries.

Handle image rendition during output generation

AEM comes with a set of default workflows and media handles to process assets. In AEM, there are pre-defined workflows to handle asset processing for the most common MIME types. Typically, for every image that you upload, AEM creates multiple renditions of the same in binary format. These renditions may be of different size, with a different resolution, with an added watermark, or some other changed characteristic. For more information about how AEM handles assets, see Processing Assets Using Media Handlers and Workflows in AEM documentation.

XML Documentation solution allows you to configure which image rendition to use at the time of generating output for your documents. For example, you can choose from one of the default image rendition or create one and use the same to publish your documents. Image rendition mapping for publishing your

documents is stored in the `/libs/fmdita/config/` **`renditionmap.xml`** file. A snippet of `renditionmap.xml` file is as follows:

NOTE: It is recommended that you create a copy of the `renditionmap.xml` file in the `apps` folder for all customizations.

```
<renditionmap>
   <mapelement>
      <mimetype>image/png</mimetype>
      <rendition output="AEMSITE">cq5dam.web.1280.1280.jpeg</rendition>
      <rendition output="PDF">original</rendition>
      <rendition output="HTML5">cq5dam.web.1280.1280.jpeg</rendition>
      <rendition output="EPUB">cq5dam.web.1280.1280.jpeg</rendition>
      <rendition output="CUSTOM">cq5dam.web.1280.1280.jpeg</rendition>
   </mapelement>
...
</renditionmap>
```

The `mimetype` element specifies the MIME type of the file format. The `rendition output` element specifies the type of output format and the name of rendition (for example, `cq5dam.web.1280.1280.jpeg`) that should be used for publishing the specified output. You can specify the image renditions to use for all supported output formats - AEMSITE, PDF, HTML5, EPUB, and CUSTOM.

If the specified rendition is not present, then XML Documentation solution publishing process first looks for the web rendition of the given image. If even the web rendition is not found, then the original rendition of the image is used.

NOTE: These image renditions control only the output generation. An image's web rendition is used when you open a document for preview or review.

Configure auto-purging period for output history

When you generate an output, the output gets created along with the output logs. For large DITA maps, these logs can take a large amount of space in your repository. By default, the logs are stored at the following location in the repository:

`/content/fmdita/metadata/outputHistory/`

Over a period of time, the collective size of all log files could run into GBs. XML Documentation solution allows you to configure a time period to keep these log files in the repository. After the specified time period, the logs along with output generation history are deleted from the repository.

NOTE: The output generation history is the log entry in the Generated Outputs list in the Outputs tab.

Configuring the history purging feature impacts the output generation for all DITA maps in the repository. In the Outputs tab of a DITA map, the history is purged after the specified number of days and at the time specified in the setting.

NOTE: Removing the log files and output generation history does not have any impact on the generated output.

Perform the following steps to set a day and time to purge output history and logs:

1) Open the Adobe Experience Manager Web Console Configuration page.

   The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

2) Search for and click on the *com.adobe.fmdita.config.ConfigManager* bundle.
3) In the **Output History Purging Period** property, specify the number of days after which the output history along with output logs are purged. By default, it is set to `5` days. If you want to disable this feature, then set this property to `0`.
4) In the **Output History Purging Time** property, specify the time when the purging process is initiated. By default, it is set to 0:00 (or 12:00 midnight). Everyday at this time, the purging process is executed on outputs generated before the number of days specified in the **Output History Purging Period** property.

   NOTE: By default, the purging feature is executed every midnight on outputs older than 5 days.
5) Click **Save**.

Change the recently generated outputs list limit

You can change the maximum number of generated outputs that are displayed in the Outputs tab for a DITA map. By default, a list of last 25 generated outputs is shown. To change the number of outputs to display in the list, update the **Outputs List Limit** setting in the `com.adobe.fmdita.config.ConfigManager` bundle.

TIP: See *Output history* for best practices around working with output history.

Output generation performance optimization

XML Documentation solution allows you to configure the output generation processes pool size that controls the number of output generation processes that run concurrently. By default, the process pool size is set to number of processing cores available in your system plus one. You might want to change this value to 1 in case you want sequential publishing. In this case, the first publishing task gets executed and the next publishing task is stored in the publishing queue.

To change the output generation processing pool size, update the **Generation Pool Size** setting in the `com.adobe.fmdita.publish.manager.PublishThreadManagerImpl` bundle.

Configure and customize workflows

Workflows enable you to automate Adobe Experience Manager (AEM) activities. A workflow consists of a series of steps that are executed in a specific order. You can define a distinct activity to execute on each step. For example, you can send an email notification to all reviewers in a group when a topic review is created. Or, send a notification to the publisher when an output generation task completes.

For more information about workflows in AEM, see:

- Administering Workflows
- Applying and participating in workflows: Working with Workflows.
- Creating workflow models and extending workflow functionality: Developing and Extending Workflows.
- Improving the performance of workflows that use significant server resources: Concurrent Workflow Processing.

The sections in this topic will walk you through various customizations that you can make in the default workflows shipped in XML Documentation solution.

TIP: See *Workflow offloading* for best practices around offloading workflow.

Customize review workflow

Every organization's content authoring team works in a specific way to meet their business requirements. In some organizations there is a dedicated editor, whereas some other organization could have automated editorial review system in place. For example, in an organization a typical authoring and publishing workflow could include tasks like - whenever an author is done with authoring content, it automatically goes to the reviewers, and when the review is complete it goes to the publisher for generating the final output. In AEM, activities that you do on your content and assets can be combined in the form of a process and mapped to an AEM workflow. For more information about workflows in AEM, see Administering Workflows in AEM documentation.

XML Documentation solution allows you to customize the default review workflow. You can use the following four custom review-related processes with your other authoring or publishing workflows.

- **Create Review**: This process prepares the metadata required to create a review task. For example, it will assign review permission to the reviewers, set the status of the topics to under review, set the review timelines, and more. Out of the four processes, this is the only mandatory process that must be included in your custom workflow. In your workflow, you may choose to include or exclude the other three processes.
- **Assign Review Task**: This process creates the review task and sends the task notification to the initiator and reviewers.
- **Send Review Email**: This process sends the review email to the initiator and reviewers.
- **Schedule Job to Close Review**: This process ensures that the review process completes on reaching the deadline.

When you are creating a custom review workflow, the first task is to set the required metadata needed by the Create Review process. To do so, you can create an ECMA script. A sample of the ECMA script that assigns the metadata is given below:

```
var workflowdata=workItem.getWorkflowData();
workflowdata.getMetaDataMap().put("initiator","admin");
workflowdata.getMetaDataMap().put("operation","AEM_REVIEW");
workflowdata.getMetaDataMap().put("orgTopics","/content/dam/xml-solution
/review.xml");
workflowdata.getMetaDataMap().put("payloadJson","{\"base\":\"/content/da
m/xml-solution\",\"asset\":[\"/content/dam/xml-solution/review.xml\"],\"
referrer\":\""}");
workflowdata.getMetaDataMap().put("deadline","2017-06-27T13:19:00.000+05
:30");
workflowdata.getMetaDataMap().put("title","Review through custom
workflow");
workflowdata.getMetaDataMap().put("description","Initiate this review
process using the AEM workflow");
workflowdata.getMetaDataMap().put("assignee","user-one", "user-two");
workflowdata.getMetaDataMap().put("status","1");
workflowdata.getMetaDataMap().put("projectPath","/content/projects/revie
w");
workflowdata.getMetaDataMap().put("startTime",
System.currentTimeMillis());
```

You can create this script in the `/etc/workflows/scripts` node. The following table describes the properties being assigned by this ECMA script:

Property	Type	Description
`initiator`	String	User ID of the user initiating the review task.
`operation`	String	A static value set as `AEM_REVIEW`.
`orgTopics`	String	Path of the topics being shared for review. Specify multiple topics separated by comma.
`payloadJson`	JSON object	Specify the following values: • *`base`*: path of the parent folder containing the topic sent for review. • `asset`: path of the topic sent for review. • `referrer`: leave it blank.
`deadline`	String	Specify the time in `yyyy-MM-dd'T'HH:mm:ss.SSSXXX` format.
`title`	String	Enter a title for the review task.

Property	Type	Description
`description`	String	Enter a description for the review task.
`assignee`	String	User ID of the users to whom you want to send the topic(s) for review.
`status`	Integer	A static value set as 1.
`startTime`	Long	Use the `System.currentTimeMillis()` function to get the current system time.

Once you have created the script, call it before calling the Create Review process in your workflow. Then, depending on your requirements, you can call the other review workflow processes.

Remove review workflow from the purge configuration

To improve the workflow engine performance, you can regularly purge completed workflow instances from the AEM repository. If you are using the default AEM configurations, then all completed workflow instances are cleaned up after a specific period of time. This also results in all review workflows to get purged from the AEM repository.

You can prevent review workflows from auto-purging by removing the review workflow model (information) from the auto-purge configuration. You need to use the **Adobe Granite Workflow Purge Configuration** to remove the review workflow models from auto-purging list.

In the **Adobe Granite Workflow Purge Configuration**, ensure that you list at least one workflow that you can safely purge. For example, you can use any of the following workflows created by XML Documentation solution:

- `/etc/workflow/models/publishditamap/jcr:content/model`
- `/etc/workflow/models/post-dita-project-creation-tasks/jcr:content/model`

Adding a workflow in the **Adobe Granite Workflow Purge Configuration** ensures that AEM purges only those workflows that are listed in the configuration. This prevents AEM from purging the review workflow information.

For more details about configuring the **Adobe Granite Workflow Purge Configuration**, see *Administering Workflow Instances* in AEM documentation.

Customize email templates

A number of the XML Documentation solution workflows make use of email notifications. For example, if you initiate a review task, an email notification is sent to the reviewers. However, to ensure that the email notification is sent, you have to enable this functionality in AEM. To enable email notification in AEM, see the article Configuring Email Notification in AEM documentation.

The XML Documentation solution contains a set of email templates that you can customize. Perform the following steps to customize these templates:

1) Log into AEM and open the CRXDE Lite mode.
2) In the Navigator tab, go to the following location:

   /libs/fmdita/mail

   NOTE: Do not make any customizations in the default configuration files available in the `libs` node. You must create an overlay of the `libs` node in the `apps` node and update the required files in the `apps` node only.
3) The mail folder contains the following customizable templates:

Template Filename	Description
`closereview.html`	This email template is used when a review task is closed.
`createreview.html`	This email template is used when a new review task is created.
`reviewapproval.css`	This CSS file contains the styling of email templates.

Customize post-output generation workflow

XML Documentation solution gives you the flexibility to specify a post-output generation workflow. You can perform some post-processing tasks on the output that gets generated using the XML Documentation solution. For example, you might want to apply some CQ tags on the generated AEM Site output, or set certain properties on the PDF output, or you might want to send an email to a set of users once the output is generated.

You can create a new workflow model to use as a post-output generation workflow. When a post-output generation workflow is triggered, the output generation workflow shares contextual information through the workflow metadata map, which you can use to perform processing on the generated output. The following table describes the contextual information shared as metadata:

Property	Type	Description
`outputName`	String	Name of the output preset used to generate the output.
`generatedPath`	String	Path in DAM where the generated output is stored.
`outputType`	com.adobe.fmdita.output.OutputType	Type of the output preset.
`outputTitle`	String	Title of the output preset.
`outputHistoryPath`	String	Repository path of the history node.

Property	Type	Description
`isSuccess`	Boolean	A flag depicting the final status of the output generation process - success or failure.
`logPath`	String	Path in DAM where the output generation logs are saved.
`generatedTime`	Long	Time at which the output generation process was triggered.
`initiator`	String	The user ID of the user who triggered the output generation workflow.

To make use of the output generation metadata, you can create an ECMA script or an OSGi bundle. A sample of the ECMA script that uses the metadata is given below:

NOTE: You can create this script in the `/etc/workflows/scripts` node.

```
var session = workflowSession.getSession(); // Obtain session object to
read/write the repository.
var payload = workItem.getWorkflowData().getPayload().toString(); // Get
the workflow payload (the ditamap file on which the generation was
triggered)
var metadata = workItem.getWorkflowData().getMetaDataMap(); // Get the
workflow metadata object
var generatedPath = metadata.get("generatedPath"); // supplied by XML
Documentation solution
var username = metadata.get("initiator"); // supplied by XML Documentation
solution
var successful = metadata.get("isSuccess"); // supplied by XML
Documentation solution
var title = metadata.get("outputTitle"); // supplied by XML Documentation
solution
var subject = "Output Generation Finished";
var message = "Generation of output " + title + " just finished " +
(successful ? "successfully. " : "unsuccessfully. ");
    message += "It was triggered by " + username;
if (successful) {
message += "<br/><br/>The path to the generated output is " +
generatedPath;
}
/*
MailerAPI.sendMail("dl-docs-authors", subject, message);
*/
```

Once you have created the script, call the custom script in your workflow. Then, depending on your requirements, you can call the other workflow processes. Once you have designed your custom workflow, call the *Finalize Post Generation* as the last step in your workflow process. The *Finalize Post Generation* step ensures that the status of the output generation task gets updated to *Finished* on completion of the output generation process. After creating a custom post-output generation workflow, you can configure it with any of your output generation presets. Select the required workflow in the *Run Post Generation Workflow* property of the required preset. When you run an output generation task using the configured output preset, the task status (in the Output tab) changes to *Post-Processing*.

Customize Update Asset workflow

By default, the *DAM Update Asset* workflow triggers whenever you create or update any AEM Asset (XML or non-XML). For example, when you create a topic or update it, the *DAM Update Asset* workflow gets executed. The *DAM Update Asset* workflow tries to extract relevant metadata from the Assets. The out-of-box *Asset Update Workflow* does not have any steps to extract any relevant metadata from a DITA file and the *DAM Update Asset* workflow generates a lot of logs at the time of execution. If you want to avoid the extra logs, you can configure the workflow to skip all XML files from processing

Perform the following steps to customize the *DAM Update Asset* workflow:

1) open the **Workflow Launchers** page.
   The default URL to access the Workflow Launchers page is:
   `http://<server name>:<port>/libs/cq/workflow/admin/console/content/launchers.html`
2) From the list of workflow launchers, open the properties of the **DAM Update Asset** workflow.
3) Add a condition with the following expression:
   `jcr:content/metadata/dc:format!=application/xml`
4) Click **Save & Close**

Configure post-processing XML workflow

XML Documentation solution creates a bunch of workflows that allow you to work with DITA content in AEM. For example, there are workflows that get executed when you upload DITA content or update existing content. These workflows parse DITA documents and perform various tasks such as setting the metadata, adding default output presets to new DITA maps, and other related tasks.

NOTE: To customize or extend the default post-processing workflows, you can use the post-processing event handler described in the *XML Documentation for Adobe Experience Manager API Reference* guide.

The following properties govern how XML Documentation solution executes the post-processing workflows:

NOTE: The following properties are accessible through the Web Console: `http://<server name>:<port>/system/console/configMgr`.

Property	Bundle Name	Description
Dynamic Outrefs	`com.adobe.fmdita.postprocess.PostProcessObservation`	For all files on which the post-processing has not been performed, it retrieves the outgoing references by parsing the topic files. It is recommended to keep this option disabled since it has a possibility of overloading the system if the number of files to be processed is large.
Post Process Threads	`com.adobe.fmdita.config.ConfigManager`	Sets the number of post-processing threads to be used for post-processing workflow. The default value is 1.

Translate content

Automate the translation of page content, assets, and user-generated content to create and maintain multilingual websites. To automate translation workflows, you integrate translation service providers with AEM and create projects for translating content into multiple languages. AEM supports human and machine translation workflows.

- Human translation: Content is sent to your translation provider and translated by professional translators. When complete, the translated content is returned and imported into AEM. When your translation provider is integrated with AEM, content is automatically exchanged between AEM and the translation provider
- Machine translation: The machine translation service immediately translates your content

Translating content involves the following steps:

1) Connect AEM with your translation service provider and create translation integration framework configurations.
2) Associate the pages of your language master with the translation service and framework configurations.
3) Identify the type of content to translate.
4) Prepare the content for translation by authoring the language master and creating the root pages of language copies.
5) Create translation projects to gather the content to translate and to prepare the translation process.
6) Use the translation projects to manage the content translation process.

When your translation service provider does not provide a connector to integration with AEM, then AEM supports the manual export and import of translated content in XML format.

TIP: See *Translation* for best practices around translating content.

Translation configurations

If the connector for the translation vendor does not support DITA content, then the component-based translation workflow needs to be enabled. Once enabled, the translatable content is sent as asset metadata. However, the connector needs to support asset metadata translation for this workflow to work.

Based on the translation workflow used in your setup, the component-based translation workflow option should be configured as follows:

1) Open the Adobe Experience Manager Web Console Configuration page.

   The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`
2) Search for and click on the *com.adobe.fmdita.config.ConfigManager* bundle.

3) Configure the **Component-Based DITA Translation Workflow** option as per your setup:
   - If you are using human translation, then *Disable* the **Component-Based Translation Workflow** option.
   - If you are using machine translation, then *Enable* the **Component-Based Translation Workflow** option.

   NOTE: If you are using translation connector, then ensure that you have configured the connector as described in the *Configuring the Translation Integration Framework* topic in AEM documentation.
4) Click **Save**.

IMPORTANT: After setting up the translation configurations, ensure that you set up the appropriate Cloud Configuration on the language folders.

Configure DITA content search

By default, AEM does not recognize DITA content, thus, it doesn't provide any mechanism to search DITA content within its repository. XML Documentation solution allows you to add the DITA content search capability in AEM repository.

Configuring DITA content search involves the following tasks:

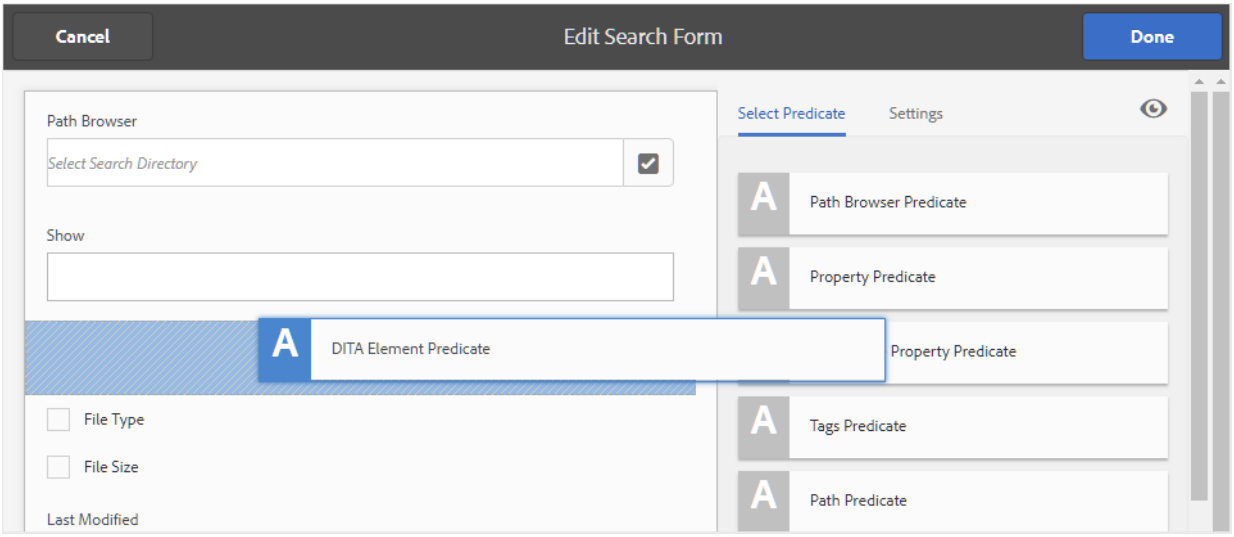
1) *Add DITA Element search component in Assets UI*
2) *Provide permissions to users*
3) *Add custom elements or attributes in search*
4) *Extract metadata from existing content*

In addition to adding search capability, you can also configure the folders that should not be included in the search. For more details, see *Exclude temporary files from search results*.

Add DITA Element search component in Assets UI

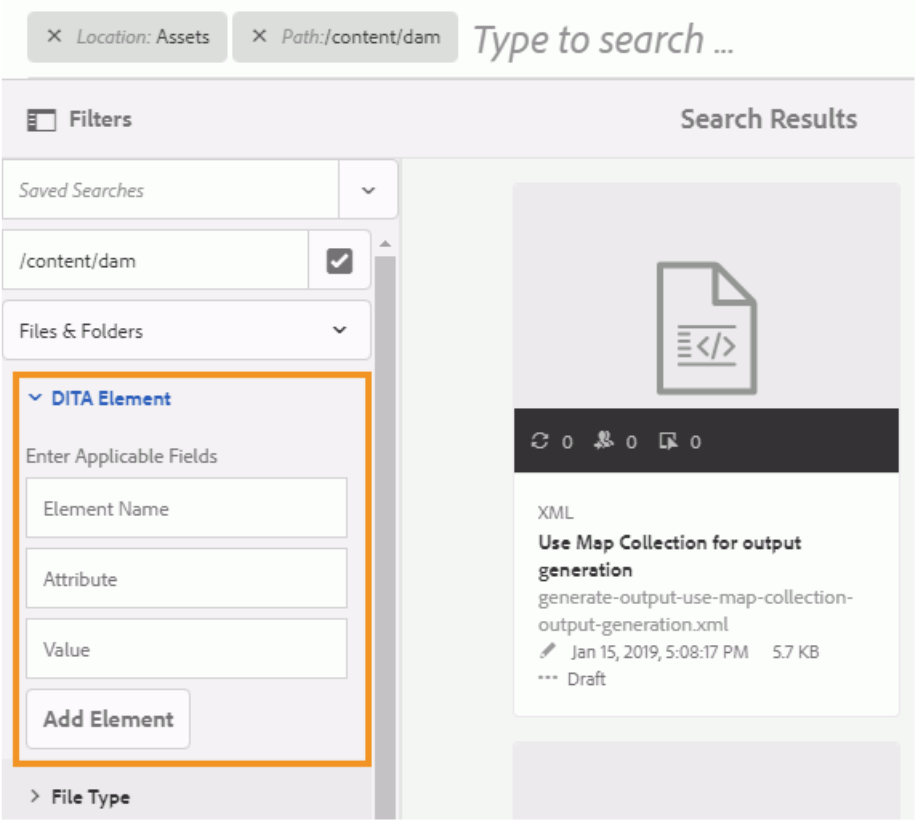
Perform the following to add DITA content search component in AEM Assets UI:

1) Log into Adobe Experience Manager as an administrator.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the **Adobe Experience Manager** link at the top and choose **Tools**.
   *(In AEM 6.1) Click on the **Experience Manager** link, and then click **Tools**.*
3) Select **General** from the list of tools and click on the **Search Forms** tile.
4) In the **Search Forms** list, select the **Assets Admin Search Rail**.
5) Click **Edit**.
6) In the **Select Predicate** tab, scroll to the end of the list.
7) Drag-and-drop **DITA Element Predicate** at the required location in the search form.



8) Click **Done** to save your changes.

When you access the Filters option in the Assets UI, you will get the DITA Element search filtering option.



Provide permissions to users

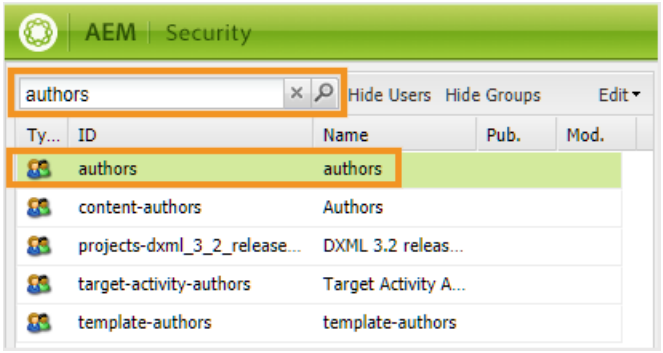
The users (Authors and Publishers) would need to be given explicit permissions to be able to access the DITA Search feature from Assets UI.

Perform the following steps to provide access to the DITA Search feature:

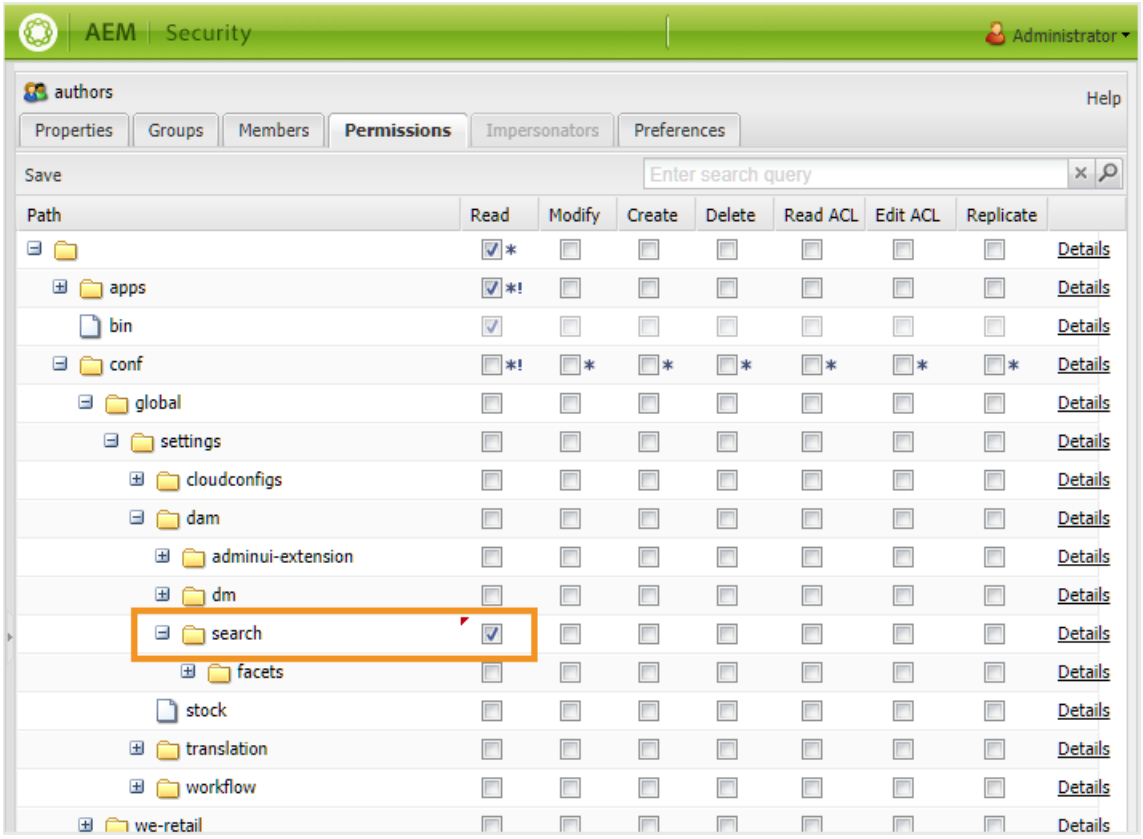
1) Access the user and group permissions page. The default URL to access the page is:

   `http://<server name>:<port>/useradmin.html`

2) Search for the user group or an individual user to whom you want to give access. For example, to give access to all users in `authors` group, enter `authors` in the **Filter Query** field and press **Enter**.



3) Select the **authors** group.
4) In the right pane, select the **Permissions** tab.
5) Navigate to the following folder location:
   `/conf/global/settings/dam/search`
6) Give the **Read** permission on the `search` folder.



7) Click **Save**.

The selected user or user group will now have access to the search DITA content feature in the Assets UI.

Add custom elements or attributes in search

For the DITA search to work, some pre-processing of the DITA content is required. This pre-processing step extracts selective content from individual DITA maps and topics so that it can be indexed for faster searching. Internally, this process is called *Serialization*. Serialization of DITA files takes place during content upload or can also be executed on-demand. It uses a configuration file to determine how much content from each DITA file should be indexed. The default location of the serialization file is:

`/libs/fmdita/config/serializationconfig.xml`

The default search configuration allows you to search for all elements and attributes within the DITA **`<prolog>`** element. If you want to search on the basis of other elements or attributes, then you need to configure the search serialization file.

NOTE: If you want to go with the default search configuration within the **<prolog>** element, then you can skip this process.

This file contains two main sections—attribute set and rule set. A snippet of the rule set section is given below:

```
<ruleset filetypes="xml dita"><!-- Element rules --><rule
xpath="//[contains(@class, 'topic/topic')]/[contains(@class,
'topic/prolog')]//*[not(*)]" text="yes" attributeset="all-attrs" /><!--
Attribute rules --><rule xpath="//[contains(@class,
'topic/topic')]/[contains(@class, 'topic/prolog')]///@[local-name() !=
'class']" /></ruleset>
```

In the rule set section, you can specify:

- Rules to extract the elements
- Rules to extract attributes

A rule consists of the following:

xpath

This is the XPath query that retrieves the elements or attributes from DITA files. The default configuration for the element rule retrieves all **<prolog>** elements. And, the default configuration for the attribute rule retrieves all attributes of **<prolog>** elements. You can specify an XPath query to serialize the elements or attributes that you want to search for.

The XPath query contains the class name of the document type. The `topic/topic` class is used for topic type DITA documents. If you want to create a rule for other DITA documents, then you must use the following class names:

Document Type	Class name
Topic	- topic/topic
Task	- topic/topic task/task
Concept	- topic/topic concept/concept
Reference	- topic/topic reference/reference
Map	- map/map

text

If you want to search for the text within the specified element, then specify the `yes` value. If you specify `no` as value, then only the attributes within the element are serialized. The attributes that you want to search for need to be specified in the attribute set section.

attributeset

Specify the ID of the attribute set that you want to associate with this rule. The value `all-attrs` is a special case to indicate that all attributes for this rule must be serialized.

An attribute set contains a list of attributes that you want to search for within DITA content. The attribute set contains the following:

id

A unique identifier for the attribute set. This `id` is specified in the `attributeset` parameter of a rule set.

attribute

A list of attributes that you want to search. For each attribute, you need to create an individual entry in the **<attribute>** element.

Perform the following steps to add custom DITA elements or attributes in the search serialization file:

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the serialization configuration file available at the following location:
   /libs/fmdita/config/serializationconfig.xml
3) Create an overlay node of the `config` folder within the `apps` node.
4) Navigate to the configuration file available in the `apps` node:
   /apps/fmdita/config/serializationconfig.xml
5) Add the required element or attribute rule sets.
6) Save the file.
7) Open the Adobe Experience Manager Web Console Configuration page. The default URL to access the configuration page is:
   http://<server name>:<port>/system/console/configMgr
8) Search for and click on the *com.adobe.fmdita.config.ConfigManager* bundle.
9) Click **Save**.

The new serialization information is stored and activated for search. However, you must extract the metadata from your existing DITA content to become available for search.

Extract metadata from existing content

Once you have made any change in the default search serialization file, you must enable the DITA Metadata Extraction option in the *com.adobe.fmdita.config.ConfigManager* bundle and then run the workflow to extract metadata. This extract the required metadata from the existing DITA files and the same is then made available for search.

In case you create new files or edit any file after updating the serialization file, then the metadata is automatically extracted from such files. The process of extracting metadata is only required for files that already exist in AEM repository.

Extracting metadata from existing DITA files involves two tasks:

1) Enabling the metadata extraction option in the configMgr
2) Running the metadata extraction workflow

Perform the following steps to enable the metadata extraction option in the configMgr:

1) Open the Adobe Experience Manager Web Console Configuration page. The default URL to access the configuration page is:

   `http://<server name>:<port>/system/console/configMgr`

2) Search for and click on the *com.adobe.fmdita.config.ConfigManager* bundle.
3) Select the **Enable DITA Metadata Extraction** option.
4) Click **Save**.

Perform the following steps to run the metadata extraction workflow:

1) Log into Adobe Experience Manager as an administrator.
2) *(In AEM 6.4, 6.3, and 6.2)* Click on the **Adobe Experience Manager** link at the top and choose **Tools**.

   *(In AEM 6.1) Click on the **Experience Manager** link, and then click **Tools**.*
3) Select **XML Documentation** from the list of tools and click the **DITA Metadata Extraction** tile.
4) If you want to extract metadata from a single file and its dependencies, click the **Select a File** link and browse for a file.
5) If you want to extract metadata from multiple files within a folder, click the **Select Folder(s)** link, browse and select the required folder. Click the **Add** button to add the folder to the serialization task list.

   NOTE: You can select and add multiple folders to a serialization task.
6) Click **Start**.
7) In the *Confirm Metadata Extraction* dialog, click **OK**.

Exclude temporary files from search results

By default, the search is performed on the entire repository of AEM. There could be some locations that you would like to exclude from search. For example, when you initiate the content translation workflow, the unapproved files remain in a temporary folder location. When you perform the search, files from this temporary location are also returned in the search results.

To prevent XML Documentation solution from searching the temporary translation folder location, you need to add temporary folder location in the exclude list.

Perform the following steps to exclude the temporary translation folder from the search:

***NOTE:** You can add any other folder location to the exclude list using this procedure.*

1) Log into AEM and open the CRXDE Lite mode.
2) Navigate to the `damAssetLucene` node available at the following location:

   `/oak:index/damAssetLucene`
3) Add the following property in the damAssetLucene node:

Property name	Type	Value
`excludedPaths`	String[]	Add the following value to this property: `/content/dam/projects/translation_output`

4) Navigate to the `lucene` node available at the following location: */oak:index/lucene*

5) Add the following property in the lucene node:

Property name	Type	Value
`excludedPaths`	String[]	Add the following values to this property: • */content/fmdita* */content/dam/projects/translation_output*

Appendix

This appendix provides best practices for working with XML Documentation for Adobe Experience Manager. Following these best practices will help you set up, organize DITA content, publish content, and develop processes around content creation, management, and publishing.

The information in this topic is intended for the following type of audiences:

- Publishers, who would run the publishing task to generate output in various formats
- Administrators, who would install and manage XML Documentation solution on Adobe Experience Manager

AEM nodes overlaid by XML Documentation solution

For customizing the user interface in AEM, XML Documentation solution uses the following `/apps` nodes:

For AEM 6.5

- `/apps/cq/core/content/nav/tools/xmladdonconfigurations`
- `/apps/cq/core/content/nav/tools/xmladdonconfigurations`
- `/apps/cq/core/content`
- `/apps/cq/inbox/extensions/itemtyperegistry/task/review-task`
- `/apps/cq/inbox/extensions/itemtyperegistry/task/approval-task`
- `/apps/cq/inbox/extensions/itemtyperegistry/task/review-task-notification`
- `/apps/cq/inbox/extensions/itemtyperegistry/task/approval-task-notification`
- `/apps/dam/gui/content/assetdetails/jcr:content/content/items/assetdetail/items/viewer`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/filter`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/editTopic`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/openinfm`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/selectmap`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/keyresolution`
- `/apps/dam/gui/content/assetdetails/jcr:content/actions/lock`
- `/apps/dam/gui/content/assetdetails/jcr:content/rails/versionHistory`
- `/apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaTopic`
- `/apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaMap`
- `/apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaTemplate`

- /apps/dam/gui/content/assets/jcr:content/actions/selection/edit
- /apps/dam/gui/content/assets/jcr:content/actions/selection/review
- /apps/dam/gui/content/assets/jcr:content/actions/selection/approval
- /apps/dam/gui/content/assets/jcr:content/actions/selection/openinfm
- /apps/dam/gui/content/assets/jcr:content/actions/selection/editTopics
- /apps/dam/gui/content/assets/jcr:content/actions/selection/delete
- /apps/dam/gui/content/assets/jcr:content/actions/selection/lock
- /apps/dam/gui/content/assets/jcr:content/actions/selection/unlock
- /apps/dam/gui/content/assets/jcr:content/actions/selection/releaselock
- /apps/dam/gui/content/assets/jcr:content/actions/primary/pasteasset2
- /apps/dam/gui/content/assets/jcr:content/header/items/deletedita
- /apps/dam/gui/content/assets/jcr:content/rails/versionHistory
- /apps/dam/gui/content/assets/assetpage
- /apps/dam/gui/content/assets/moveassetwizard/jcr:content/head/clientlibs
- /apps/dam/gui/content/assets/moveassetwizard/jcr:content/body/items/form/items/wizard/items/ditaStep
- /apps/dam/gui/content/commons/availablecolumns
- /apps/dam/gui/coral/components/admin/contentrenderer/row/asset/asset.jsp
- /apps/dam/gui/coral/components/admin/customsearch
- /apps/settings/dam/search/facets
- /apps/dam/gui/content/nav/navigationlinks

For AEM 6.4 and 6.3

- /apps/cq/core/content
- /apps/cq/core/content/nav/tools/xmladdonconfigurations
- /apps/cq/inbox/extensions/itemtyperegistry/task/review-task
- /apps/cq/inbox/extensions/itemtyperegistry/task/approval-task
- /apps/cq/inbox/extensions/itemtyperegistry/task/review-task-notification
- /apps/cq/inbox/extensions/itemtyperegistry/task/approval-task-notification
- /apps/dam/gui/content/assetdetails/jcr:content/content/items/assetdetail/items/viewer
- /apps/dam/gui/content/assetdetails/jcr:content/actions/filter
- /apps/dam/gui/content/assetdetails/jcr:content/actions/editTopic
- /apps/dam/gui/content/assetdetails/jcr:content/actions/openinfm
- /apps/dam/gui/content/assetdetails/jcr:content/actions/selectmap
- /apps/dam/gui/content/assetdetails/jcr:content/actions/keyresolution
- /apps/dam/gui/content/assetdetails/jcr:content/actions/lock

- /apps/dam/gui/content/assetdetails/jcr:content/rails/versionHistory
- /apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaTopic
- /apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaMap
- /apps/dam/gui/content/assets/jcr:content/actions/secondary/create/items/ditaTemplate
- /apps/dam/gui/content/assets/jcr:content/actions/selection/edit
- /apps/dam/gui/content/assets/jcr:content/actions/selection/review
- /apps/dam/gui/content/assets/jcr:content/actions/selection/approval
- /apps/dam/gui/content/assets/jcr:content/actions/selection/openinfm
- /apps/dam/gui/content/assets/jcr:content/actions/selection/editTopics
- /apps/dam/gui/content/assets/jcr:content/actions/selection/delete
- /apps/dam/gui/content/assets/jcr:content/actions/selection/lock
- /apps/dam/gui/content/assets/jcr:content/actions/selection/unlock
- /apps/dam/gui/content/assets/jcr:content/actions/selection/releaselock
- /apps/dam/gui/content/assets/jcr:content/actions/primary/pasteasset2
- /apps/dam/gui/content/assets/jcr:content/header/items/deletedita
- /apps/dam/gui/content/assets/jcr:content/rails/versionHistory
- /apps/dam/gui/content/assets/assetpage
- /apps/dam/gui/content/assets/moveassetwizard/jcr:content/head/clientlibs
- /apps/dam/gui/content/assets/moveassetwizard/jcr:content/body/items/form/items/wizard/items/ditaStep
- /apps/dam/gui/coral/components/admin/contentrenderer/row/asset/asset.jsp
- /apps/dam/gui/content/nav/navigationlinks

Custom templates

- Custom topic templates can be added in /content/dam/dita-templates/topics folder. Custom DITA map templates can be added in /content/dam/dita-templates/maps folder. Ensure that you associate your custom template with the global or folder-level profile, otherwise it won't be available for authoring.
- It is recommended to set the Title and Description properties of a template that will help in easily identifying the template in the DITA map or topic creation page. These properties can be set using the Properties page in the Assets UI.

Content migration and upload

Consider the following best practices while migrating content to AEM from third-party DITA authoring tools, unstructured content from FrameMaker, or directly uploading DITA content to AEM.

- After uploading content, it is recommended to let the post-processing workflows complete before triggering publish or any other heavyweight task.
- It is important not to disable the post-processing workflows or their launchers.
- When you upload an asset in AEM, AEM launches a workflow to extract metadata or create renditions. Therefore, when you migrate a large amount of content, do it in batches of 500 with 2-5 minutes of interval in-between. The interval between the workflows reduces the load on the system.
- Post-processing workflows also create the default set of output presets on DITA maps. Also, automatic links fixing also relies on the post-processing workflow.
- When a DITA map without any output presets is added to a map collection, it appears blank on the page.
- Set Document State workflow sets the document state of new DITA map or topic content as Draft just after uploading or creating the content. This also happens if you are copying or moving DITA content. Disabling it causes document state of new files not set to the Draft status immediately.
- When moving, copying, or renaming DITA content, the internal DITA references are adjusted only if the movement is triggered through the Move/Copy option in the DAM UI. If you move, copy, or rename DITA content using the JCR API, or through CRXDE, or any other API, then you must explicitly fix any references to or from the moved or copied content.
- When same content is uploaded to same location on AEM assets via WebDAV or FrameMaker, then the content gets automatically versioned. This can lead to repository bloat in cases where users don't wish to keep older versions. AEM Assets' version purging should be configured in such cases.
- DITA map creation in FrameMaker to DITA conversion works only for FrameMaker Publishing Server with FrameMaker (2017 release) Update 1 and later builds.
- Use in-band metadata wherever metadata is defined as inline attribute-value pairs.

NOTE: Elements metadata node gets created only till XML Documentation solution build 2.1. This feature will be removed from future builds. Users upgrading to newer builds from 2.1 or later will have to use a script to get rid of existing elements metadata.

Prepare InDesign documents and mapping files for conversion

Prepare InDesign files for conversion

InDesign provide authors with a rich set of features for creating attractive and complex documents. Often this means that the various parts of a document are placed on the page visually, but with no attempt to provide any flow between those text frames. When the '*reading order*' of the text frames is not defined, the IDML file will contain stories that may not follow any meaningful order. The end result will be one or more DITA topics with paragraphs, tables and graphics in a random order.

While it is possible to edit the DITA content into a sensible order in a DITA editor, it is much easier to fix the InDesign file before creating the IDML file. This can be done without altering the look of the source document. It also has the benefit of making the source document accessible by properly defining the reading order.

Threading text frames

InDesign uses the term *'threading'* for the process of linking one frame to another. For more details about threading text frames, see *Threading text* topic in InDesign documentation.

Overlapping frames

Some InDesign documents use un-threaded overlapping frames for layout reasons. It may be very difficult to merge this content into the main thread. The best option may be to edit the result in the DITA environment.

InDesign stories

Each threaded flow of content in an InDesign document is a called a '*story*'. For best results, it is recommended to keep the number of stories limited. However, there are some parts of your document that may not be needed in the DITA output. For example, page footers are rarely need, but may appear in the middle of a topic if they are not handled carefully.

The easiest way to exclude text that is not required in the document is to give it a special *Paragraph Tag* that is only used for the unwanted content. For example, instead of reusing a *[Basic Paragraph]* for the footer, create a dedicated *Footer* tag. Then in the `MapStyle` file simply set the *Footer* paragraphs to be dropped like this:

```
<paraRule style="Footer" local="0" refactor="drop">
   <attributeRules createID="false"/>
</paraRule>
```

Mapping to DITA doctypes

It is essential that your source document has at least one Paragraph style or Element that can mark the start of a topic. It is common for documents to use *Heading1* as the name of the top-level titles in the document. You can then create a mapping from that style to a specific DITA doctype. If your document is well organized and the use of *Heading1* is constant throughout, then you will get good results.

Multiple DITA doctypes

If some of the *Heading1* paragraphs are need to convert to different DITA doctypes, then duplicate the paragraph style in InDesign. Give these styles an easy to recognize name such as *Heading1_genTask* or *Heading1_troubleshooting* as appropriate. Then, setup the `mapStyle` file as shown below:

```
<doctypes>
   <doctypeParaRule style="Heading1" local="0" mapToDoctype="concept">
      <attributeRules createID="true"/>
   </doctypeParaRule>
   <doctypeParaRule style="Heading1_genTask" local="0" mapToDoctype="
generalTask">
      <attributeRules createID="true"/>
   </doctypeParaRule>
   <doctypeParaRule style="Heading1_troubleshooting"
local="0"mapToDoctype=" troubleshooting">
```

```
        <attributeRules createID="true"/>
      </doctypeParaRule>
    </doctypes>
```

Structured InDesign documents

InDesign has a loose relationship with XML. While a document can include an XML DTD and the main story may be valid against that DTD, it is also possible to create hybrid documents where some of the content is XML, but no DTD is included. These are the undesirable cases for a successful conversion to DITA. If a document contains XML parts, try to save the output to XML and see if the results are acceptable. If not, then DITA content will also include invalid content, or may fail completely.

Table formatting

The conversion from InDesign table formatting rules to the equivalent table formatting in DITA is a complex process. This is due to the rich formatting features available in the source files compared to the basic options provided by the Oasis (CALS) table model used in DITA. Vertical and horizontal text alignment is provided and gives similar results although Justified text is always justified according to the text direction, while InDesign allows Left Justified and Right Justified.

InDesign's handling of column and row separators is again far more capable than the Oasis table model's basic options. InDesign provides four cell borders—Border type (solid or pattern), Border weight, Border color, Border tint, Border gap color and border gap tint. All of these have to be mapped down to borders on the right and bottom of each cell (entry element) where the only choices are 0 or 1 – hide the border or show the border.

Border ruling in InDesign can be applied at the following levels:

- Table Styles
- Cell Styles
- Local overrides on each cell

The InDesign to DITA conversion process applies the border ruling as follows:

- Table Styles are mapped to the `colspec/@colsep` attribute for vertical rules. Horizontal rules are mapped to the `row/@rowsep` attribute. In both cases, if the border is not defined, then the attribute is not created.
- Cell Styles are mapped to the `entry/@colsep` and `entry/@rowsep` attributes. These values will override any Table Style derived border ruling.
- Local overrides apply the formatting directly to the cell and override Table Styles and Cell Styles.

Alternating patterns

InDesign Table Styles allow column and cell ruling to follow an alternating pattern. While this feature is supported for conversion, the results will only be obvious when one pattern group maps to show ruling (1) and the other pattern group maps to hide ruling (0).

Prepare the mapping file for InDesign to DITA migration

The correct DITA conversion requires a mapping file that matches the contents of the source document. For unstructured InDesign documents, this means all available Paragraph Styles and Character Styles

need to be mapped. For XML structured InDesign documents all elements in its associated DTD must be mapped.

The mapping files for unstructured and structured InDesign documents are different. This is due to the more complex processing requirements for converting unstructured source content to DITA.

A sample of the mapping file is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE styleMap SYSTEM "mapStyle.dtd">
<styleMap xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mapStyle.xsd" >
   <doctypes>
      <mapDoctypeParaRule root="itpx:stories" mapToDoctype="map">
         <attributeRules createID="true">
            <addNew name="outputclass" value="map"/>
         </attributeRules>
      </mapDoctypeParaRule>
     <doctypeParaRule style="Heading 1" local="0" mapToDoctype="concept">
         <attributeRules createID="true"/>
      </doctypeParaRule>
      <doctypeParaRule style="Heading A" local="0" mapToDoctype="topic">
         <attributeRules createID="true"/>
      </doctypeParaRule>
   </doctypes>
   <wrappingRules>
      <wrap elements="li+" context="number" wrapper="ol">
         <attributeRules createID="true"/>
      </wrap>
      <wrap elements="li+" context="bullet" wrapper="ul">
         <attributeRules createID="true"/>
      </wrap>
   </wrappingRules>
   <paragraphStyleRules>
      <paraRule style="Heading 2" local="0"  mapTo="p">
         <attributeRules createID="true"/>
      </paraRule>
      <paraRule style="Heading 3" local="0"  mapTo="p">
         <attributeRules createID="true"/>
      </paraRule>
      <paraRule style="List Paragraph" local="p[-|-|-|-|-|b|-|-]"
context="bullet" mapTo="li">
         <attributeRules createID="true"/>
      </paraRule>
      <paraRule style="List Paragraph" local="p[-|-|-|-|-|n|-|-]"
context="number" mapTo="li">
         <attributeRules createID="true"/>
      </paraRule>
      <paraRule style="List Paragraph" local="0" context="bullet"
mapTo="li">
         <attributeRules createID="true"/>
```

```
      </paraRule>
      <paraRule style="Normal" local="0"  mapTo="p">
         <attributeRules createID="true"/>
      </paraRule>
     <paraRule style="Normal" local="p[-|-|-|-|-|b|-|-]" context="bullet"
mapTo="li">
         <attributeRules createID="true"/>
      </paraRule>
      <paraRule style="Title" local="0"  mapTo="p">
         <attributeRules createID="true"/>
      </paraRule>
   </paragraphStyleRules>
   <characterStyleRules>
      <charRule style="Bold" local="0" mapTo="b">
         <attributeRules createID="false"/>
      </charRule>
      <charRule style="Code" local="0" mapTo="codeph">
         <attributeRules createID="false"/>
      </charRule>
     <charRule style="[No character style]" local="c[Bold|-|-|-|-|-|-|-]"
mapTo="b">
         <attributeRules createID="false"/>
      </charRule>
      <charRule style="[No character style]"
local="c[Italic|-|-|-|-|-|-|-]" mapTo="i">
         <attributeRules createID="false"/>
      </charRule>
   </characterStyleRules>
</styleMap>
```

The mapping file is an XML file with a simple structure that lists all the source paragraph styles and character style codes. The file contents are explained below:

Style Mapping

In the **<styleMap>** element, you can specify two optional attributes – ***@map_date*** and ***@map_version*** for recording the version of the mapping file.

Document Type

The **<doctypes>** element lists the supported DITA map and topic mappings.

Map document type paragraph rules

The **<mapDoctypeParaRule>** element is mandatory. This element's attributes must not be edited because the source XML's root element is always mapped to the DITA map's root **<map>** element.

Document type paragraph rule

The **<doctypeParaRule>** element is mandatory. This gives the conversion process a way to identify the start of a new topic. Normally the ***@style*** attribute is be used on its own with the ***@local*** attribute set to `0`. However, if there are always local formatting overrides on the chosen style, you will have to add

a rule for each style plus its local overrides. This is simple to recognize in the generated mapping file where it would be possible to find this or similar:

```
<paraRule style="Heading 1" local="0" mapTo="p">
   <attributeRules createID="true"/>
</paraRule>
<paraRule style="Heading 1" local="p[Italic|-|-|-|-|-|-]" mapTo="p">
   <attributeRules createID="true"/>
</paraRule>
```

In the above example, there are two **<paraRule>** elements for ***@style*** = "Heading1". Simply, create an equivalent **<doctypeParaRule>** elements with the ***@mapToDoctype*** attribute set as required.

The attributes used in the **<doctypeParaRule>** are explained below:

- ***@style***: The name of a style in the source InDesign document.
- ***@local***: See *Local formatting codes*.
- ***@mapToDoctype***: The name of a DITA topic type from an enumerated list of all valid **<doctypes>**.

Element wrapping rules

The element wrapping rules define the ways to wrap or move elements in the incoming document into a predefined element according to a set of attribute values.

<wrap> *element*

This is an optional element. The **<wrap>** element lists the elements that will be wrapped or moved. Wrapping is typically used where a series of elements must be given a common parent element. For example, multiple **<li>** elements being wrapped in a **<ol>** element. Additionally, **<wrap>** can be used for moving elements such as titles for figures and tables.

The attributes used in the **<wrap>** are explained below:

- ***@element***: A plus sign after an element name shows that all adjacent elements with the same name will be wrapped in the element named in the ***@wrapper***attribute.
- ***@wrapper***: The name of the wrapping element.
- ***@context***: Provides a way to further refine how a given element is wrapped. The following example shows a way to map a series of **<li>** elements in either an ordered list **<ol>** or an unordered list **<ul>** according to the ***@context*** value (the context is defined on the **<paraRule>** element):

```
<wrap elements="li+" context="number" wrapper="ol">
   <attributeRules createID="true"/>
</wrap>
<wrap elements="li+" context="bullet" wrapper="ul">
   <attributeRules createID="true"/>
</wrap>
```

The following example shows how to create a **<fig>** element from a **<title>** and an **<image>** element:

- ***@elements***: The elements listed and separated by a comma will be wrapped in the element named in the ***@wrapper*** attribute. Due to the common practice of including figure titles below the image, the title will be the **<title>** element immediately following the **<image>**.

  The following wrap rule:

  ```
  <wrap elements="title, image" context="FigTitle" wrapper="fig">
     <attributeRules createID="true"/>
  </wrap>
  ```

  Converts the following intermediate XML:

  ```
  <image href="Links/myImage.png" scale="59">
     <title>IDML2DITA workflow</title>
  ```

  Into the following valid DITA figure structure:

  ```
  <fig id="id397504">
     <title>IDML2DITA workflow</title>
     <image href="Links/myImage.png" scale="59">
  </fig>
  ```

- ***@wrapper***: The name of the wrapping element.
- ***@context***: Provides a way to further refine how a given element is wrapped (the context is defined on the **<paraRule>** element).

The following example shows how to move a **<title>** into a **<table>**:

- ***@elements***: The **<title>** element that is located either immediately before or immediately after a **<table>** will be wrapped in the element named in the ***@wrapper*** attribute. An XPath-style predicate can identify the position of the title element as `[before]` or `[after]`.

  Example: The following wrap rule:

  ```
  <wrap elements="title[before]" context="TableTitle" wrapper="table">
     <attributeRules createID="true"/>
  </wrap>
  ```

  Converts the following intermediate XML:

  ```
  <title>IDML2DITA workflow</title>
  <table id="id289742" outputclass="BasicTable">
     <tgroup cols="2">
        <colspec colname="0" colwidth="0.7*">
           <colspec colname="1" colwidth="0.3*">
  ```

  Into this valid DITA figure structure:

  ```
  <table id="id289742" outputclass="BasicTable">
     <title>IDML2DITA workflow</title>
     <tgroup cols="2">
        <colspec colname="0" colwidth="0.7*">
           <colspec colname="1" colwidth="0.3*">
  ```

- ***@wrapper***: The name of the wrapping element.
- ***@context***: Provides a way to further refine how a given element is wrapped (the context is defined on the **<paraRule>** element).

Paragraph style rules

The **<paragraphStyleRule>** elements are described below:

<paraRule> ***element***

The **<paraRule>** element is mandatory. This specifies the mapping rules for all Paragraph Styles. In an InDesign document, all text is contained within sub-structure of Paragraph Styles, even paragraphs without any style are named `[No paragraph style]`. The square brackets, these indicate a built-in InDesign style name.

NOTE: The square brackets indicate a built-in InDesign style name.

The attributes used in the **<paraRule>** are explained below:

- ***@style***: The name of a style in the source InDesign document.
- ***@local***: See *Local formatting codes*.
- ***@mapTo***: The name of a DITA target element.
- ***@context***: This attribute is used to link to a specific **wrap** rule when more than one wrapper choice is available. Example: the **<li>** element may be wrapped in either an **<ol>**, or a **<ul>** element. To identify the different list types, you may use a specific style name or the ***@local*** attribute which can show the following:
  - `local="p[-|-|-|-|-|b|-|-]"` Where the 'b' in field 6 indicates a bulleted list item. In this case set ***@context*** to '`bullet`'.
  - `local="p[-|-|-|-|-|n|-|-]"` Where the 'n' in field 6 indicates a numbered list item. In this case set ***@context*** to '`number`'.
- ***@commentOut***: This attribute enables the wrapping of the target element in XML comments so that the information is not lost but can be handled manually by the user. This is useful if the source content cannot be forced to conform to DITA structure rules.
- ***@refactor***: This optional attribute has a choice of two values:
  - `unwrap`: The matched element is removed while retaining its content.
  - `drop`: The matched element and all its content is removed.

Character style rules

The **<charRule>** elements are described below:

NOTE: There will be no mapping for the built-in character style `[No character style]` when `local="0"`, because they are removed during the preprocessing.

<charRule> ***element***

This is an optional element.

These are the mapping rules for all Character Styles. In an InDesign document, all text is contained within child elements of Character Styles.

The attributes used in the **<charRule>** are explained below:

- ***@style***: The name of a style in the source InDesign document.
- ***@local***: See *Local formatting codes*.
- ***@mapTo***: The name of a DITA target element.
- ***@refactor***: This optional attribute has a choice of two values:

- `unwrap`: The matched element is removed while retaining its content.
- `drop`: The matched element and all its content is removed.

Attribute rules

This element can be a child of the following element contexts:

- **<mapDoctypeParaRule>**
- **<mapDoctypeElemRule>**
- **<doctypeParaRule>**
- **<doctypeElemRule>**
- **<paraRule>**
- **<charRule>**
- **<elementRule>**

The purpose of attribute rules is to manage the attributes for the matched elements.

Depending on the context, the following attributes are available the **<attributeRules>** element:

- ***@createID***: Generates a unique ID for the matching elements. Allowed values `true` or `false`. Available in all contexts.
- ***@copyAll***: Copies all attributes from the source XML content for structured source files only. Allowed values are `true` or `false`. Available for contexts **<mapDoctypeParaRule>**, **<mapDoctypeElemRule>**, **<doctypeElemRule>** and **<elementRule>**.

The attributes used in the **<attributeRules>** are explained below:

NOTE: This element can contain multiple child elements.

- <**addNew>**: Adds a new attribute to the matched element. Available for all contexts. It has two attributes:
  - ***@name***: Must be a legal XML name, preferably valid for the DITA context.
  - ***@value***: Can be literal text or a simple XPath expression.
- **<copyAtt>**: Copies a single attribute to the target while optionally renaming it in the process. The value is not changed. Available for contexts **<mapDoctypeParaRule>**, **<mapDoctypeElem-Rule>**, **<doctypeElemRule>** and **<elementRule>**. When this element is present the ***@copyAllAtts*** value is assumed to be `false`. It has two attributes:
  - ***@name***: Must be the name of an attribute present on the source XML element.
  - ***@mapTo***: Must be a legal XML name, preferably valid for the DITA context.

Local formatting codes

In any InDesign document, it is possible for paragraph styles and character styles to carry several hundred different formatting overrides. Most of these properties provide no useful role in the conversion process. However, we have identified a core set of formatting features that do affect the semantics of the document and need to influence the conversion process.

The ***@local*** attributes are presented as a special delimited format where eight fields are provided along with a prefix to show the type of formatting override. The formatting codes fields are listed below:

- Prefix **p** for para style local override or **c** for character style local override.

- **Font style** which is the family name and properties such as '***Bold Condensed Italic***'.
- **Font size** in points.
- **Character position** for superscript or subscript.
- **Under** for underscore.
- **Strike** for strikethrough.
- **List code** to identify list type as bulleted or Numbered – not always used by InDesign.
- **Bullet code** lists all defined bullet types in the document.
- **Number code** lists all defined numbering styles in the document.

Careful use of this feature allows otherwise lost local formatting can help to improve the quality of a transfer from styled content into DITA. This example can be resolved to mean Italic, 16pt text in a bulleted list: `p[Italic|16|-|-|-|b|-|-]`.

Structure Mapping

The structure mapping file is similar to the Style Mapping file with a simple structure that lists all the source element and relevant attribute types. Two attributes, ***@map_date*** and ***@map_version*** are provided for recording the version of the mapping file to be used.

Document Type

The **<doctypes>** element lists the supported DITA map and topic mappings.

Map document type element rules

The **<mapDoctypeElemRule>** element is mandatory. This element's attributes must not be edited because the source XML's root element is always mapped to the DITA map's root **<map>** element.

Element wrapping rules

See **Element wrapping rules**.

<elementRules> element

This lists all **Paragraph style rules**elements.

<elementRule> element

The **<elementRule>** element is mandatory. These are the mapping rules for all source elements. While an InDesign document does contain unstructured style elements, these are ignored for structured content unless the '***hybrid mode***' processing is enabled.

The attributes used in the **<elementRule>** are explained below:

- ***@elementName***: The name of a element in the source InDesign document.
- ***@local***: See *Local formatting codes*. (Only useful for Hybrid documents).
- ***@mapTo***: The name of a DITA target element.
- ***@refactor***: This optional attribute has a choice of two values:
  - `unwrap`: The matched element is removed while retaining its content.
  - `drop`: The matched element and all its content is removed.

- ***@context***: This attribute is used to link to a specific wrap rule when more than one wrapper choice is available. Example: the **<li>** element may be wrapped in either an **<ol>**, or a **<ul>** element.
- ***@commentOut***: This attribute enables the wrapping of the target element in XML comments so that the information is not lost but can be handled manually by the user. This is useful if the source content cannot be forced to conform to DITA structure rules.

Workflow offloading

XML Documentation solution post-process and publish workflows do not support the traditional AEM model of workflow off-loading using sling job queues. However, load sharing can still be achieved by having separate instances for asset upload, authoring, and production instances.

- **Asset upload instance**: is where all assets are uploaded and post-processed. Replication agents can be used to replicate processed assets (even DITA content) to the authoring instance.
- **Authoring instance**: is where all authoring takes place. Replication agents can be configured to replicate reviewed and approved content to the production instance to be published.
- **Production instance**: is where the publishing takes place. Depending on requirements this can be replaced with a publishing instance which takes care of publishing the DITA content and replication agents replicate the published pages to a different server which then acts as production.

DITA-OT Profiles configuration

The following guidelines are for DITA-OT, which can be configured using the DITA Profiles (Adobe Experience Manager link > XML Documentation > Profiles) or the Configuration Manager.

Timeout

The DITA-OT timeout value defines how long XML Documentation solution waits for a DITA-OT child process to complete before considering it in deadlock. If you are publishing large documents, you may need to increase this value. You can tell if XML Documentation solution is timing out while waiting for DITA-OT by looking at the output generation log file. For more information on accessing the log files, see the *View and check the log file* section in the XML Documentation for Adobe Experience Manager User Guide. This property can be configured in the Profile.

Memory requirements

The amount of memory required by DITA-OT depends on a lot of factors like the size of content, nested references, number of files, and more. If you observe publishing failures with the output generation logs containing an `OutOfMemoryException` in DITA-OT stack trace, you will need to increase the amount of memory allocated to DITA-OT. For more details on how to change memory allocation from DITA-OT command-line prompt, see Increasing Java memory allocation. This property can be configured in the Profile.

Generation Pool Size

By default, the **Generation Pool Size** is set to the number of processors available to the Java Virtual Machine +1. This setting is available in the *com.adobe.fmdita.publish.manager.PublishThreadManagerImpl* component in the Configuration Manager.

NOTE: The **Generation Pool Size** configuration only controls how many concurrent external DITA-OT processes are launched and not the actual number of threads taken up by those DITA-OT processes. External from XML Documentation solution's perspective includes both DITA-OT and FrameMaker Publishing Server tasks. For example, if thread pool configuration is set to 3, then at any given time a maximum of 3 concurrent DITA-OT processes will be active. Each of these DITA-OT processes can launch any number of further threads.

Using custom DITA-OT

XML Documentation solution provides an option to use custom DITA-OT package in place of the default package shipped with XML Documentation solution. Consider the following points while using a custom DITA-OT package:

- XML Documentation solution supports DITA 1.2 and 1.3 document types. Specialized DITA is also supported, however it is recommended to use the base DITA specifications. Use specialized DITA only if your content needs are not met with the base DITA document types.
- The default DITA-OT package shipped with XML Documentation solution contains a plug-in to generate EPUB output. This plug-in is not available in the DITA-OT package available for download from the official DITA-OT website. If you are using a custom DITA-OT package, ensure that it contains the plug-in to generate EPUB output, else the EPUB output generation will fail.
- The PDF output preset uses the `pdfx` transformation that is supported by the default DITA-OT package. When using custom DITA-OT package, you must specify the transformation that should be used to generate the PDF output. Also, this transformation has to be supported by your custom DITA-OT plug-in.

Export custom DITA profile as package

You can export your custom DITA profiles as a package and then import it to the other instances of XML Documentation solution. This saves time when you have multiple instances of XML Documentation solution and want to have same DITA profile on all of them.

Perform the following steps to use your custom DITA profile on other XML Documentation solution instances:

1) Customize your DITA profile and save it.
2) Do the following to create a package:
    a) Open CRX package manager and click **Create package**.
    b) Enter package name and click **OK**.
    c) Click **Edit** to edit the package.

d) On the **Edit Package** dialog, select **Filters** > **Add filter**.

e) Select a profile from the root path `/content/fmdita/profiles/` and click **Done**.

NOTE: You can check the name of the profile from the profile properties.

f) Click **Save** to save your changes.

g) Click **Build** to create the package.

3) After your package is created, click **Download** to download the package to your local machine.

4) Import the downloaded package to the other instances of XLM Documentation solution using the package manager.

5) Once imported, open `com.adobe.fmdita.config.ConfigManager` and save it without any modifications at `http://<server-url>:<port>/system/console/configMgr` to enable the installed DITA profile.

For more information on how to work with a package, see Packaging Adobe Experience Manager 6 applications in AEM documentation.

Output history

- Every output generation information is tracked as an entry under the following repository node: `/content/fmdita/metadata/outputHistory`
- If you observe that the output history tab in the DITA map console is empty even after you have generated multiple outputs, it could mean that your output history node is corrupted. You can clean the output history for a particular DITA map file by deleting its output history node.

  Note: Deleting the output history node for a DITA map deletes its complete output history.
- Output history nodes are named on the UUIDs of their corresponding DITA map files. You can find the output history node for a particular DITA map by performing the following steps:

  a) Using CRXDE, navigate to the DITA map file node.

  b) Find the `jcr:uuid` property on the DITA map node and copy its value.

  c) Navigate to `/content/fmdita/metadata/outputHistory/<value_you_copied_in_previous_step>`.

  d) The selected node is the output history node for the concerned DITA map.
- The output history nodes also store the output generation logs. Currently there is no automated way to clear old output history. However, user can write custom scripts to delete old output history nodes by checking `jcr:created` property of logs node.

PDF generation

When creating PDF output and storing them in DAM, you should take care that asset update workflow triggers a sub-asset creation for the generated PDFs. These sub-assets take up a lot of disk space and the

sub-asset creation workflow also takes up a lot of CPU. In most of the cases, you won't have a need of these sub-assets and should preferably disable these.

Using post-generation workflows

For any custom post-generation workflow, you must put in the **Finalize Post Generation** step as the last step. Otherwise, the output generation will not be properly tracked in output history.

Unstructured publishing

- The custom `.sts` file for HTML publishing should be of the same FrameMaker version as that of the FrameMaker Publishing Server.
- The `joboptions.xml` file located at `/libs/fmdita/config/` should not be empty. This file specifies distiller job options to be used for PDF creation of unstructured files.

Baseline

- Do not remove a baseline that is used in any output preset. Trying to generate output using a deleted baseline would result in a failure.
- If different versions of an asset are referenced in different topics and if that referenced file is picked automatically, the latest version of the referenced file will be used while publishing. Make sure that same version for such asset is referenced everywhere to avoid undesirable results in published outputs.
- A DITA map containing baseline should not be moved to some other location. Doing so results in loss of baseline information from the DITA map.

AEM Site publishing

Page naming

Till XML Documentation solution version 2.0, all generated site page names were in lowercase. Starting from XML Documentation solution version 2.1, site page names have same case as the corresponding DITA filename. So anyone upgrading from 2.0 to a later version might need to modify external links to their site pages to take care of case changes.

Activation to publish instance

When AEM Site output is generated with **Delete and Create** option selected for the **Existing Output Pages** setting, then the previously generated output is first deleted and then pages are re-created. If replication agent is configured on author instance to automatically replicate all changes to AEM site pages to publish instance, then it is possible that by the time pages get re-created, page deletion event is propagated on the publish instance by the replication agent. In that case, the AEM site pages will be marked **deactivated** on the publish instance.

AEM Site page creation logic

If you are re-publishing some content after changing the **Design Path** setting to a different template, it is recommended to either publish to a different destination or select the **Delete and Create** setting to avoid inconsistent output.

AEM Site templates

You can customize the site template to control the structure of the created pages and also the existence of the default search and landing pages, see *Customize AEM Site output design template for more information*.

Concurrent publishing of the same DITA map

This should be done only at different destination paths. If there are two or more publishing workflows writing to a common destination path, some or all of them might fail and the consistency of the published content is not guaranteed.

Guidelines on creating and overwriting output

When you publish your output in AEM Site format, then you have an option to manage existing pages in the destination. When you select **Overwrite Content** option in the **Existing Output Pages** setting, then for any existing page in the destination, XML Documentation solution recreates its `contentnode` and `headnode`. Then the new content is written in the newly created nodes. This option does not create any new revision of page.

The **Delete and Create** option in the **Existing Output Pages** setting deletes any existing pages in the destination path and then publishes the new content.

Sub-node publishing

XML Documentation solution supports non-destructive publishing, so that you can combine the published DITA content with content created through other means in a page. For this to work, it is important that your DITA map structure mirrors your site structure and the topic filenames match the page names (case-sensitive since version 2.1). Once you have your site structure and the corresponding DITA map hierarchy ready, you can configure the AEM Site template for this purpose:

- Use the `topicContentNode` and `topicHeadNode` properties of your AEM Site template node to specify the node paths where you want DITA content to be published.
- During publishing, XML Documentation solution cleans these nodes of any existing content and publishes new content in them without touching any other nodes in the page structure.
- You must select the **Overwrite Content** option in the **Existing Output Pages** setting in AEM Site preset so that XML Documentation solution does not delete and recreate the whole page.
- For more details about the Site template node and its properties, see *Customize AEM Site output design template*.

Other recommendations

This topic contains information about various other tasks and workflows that you can optimize.

Permissions

To give a user permissions to use the XML Documentation solution's all capabilities and generate output in a specific format, you can add that user to the `Publishers` group created by XML Documentation solution. This ensures that the user has full access to all XML Documentation solution publishing functionalities.

In addition to above permissions, read/write access to specific DAM folders can be configured for this user as per your requirements from the **useradmin** interface.

Translation

- All translatable content should be placed under a valid self-contained language folder (language copy root) following the naming convention appropriate to the AEM version. For example, if there is a folder for keeping German content, then there should be no content reference from German folder to any other language folder.
- Component-based translation can be used even when the translation vendor does not support XML/DITA translation. However, asset metadata translation needs to be supported by the translation vendor. The Component-based translation is recommended only for machine-based translation process. For human translation, disable the Component-based translation option in the Config Manager.
- Any common content that does not require translation should be stored outside of any language folder. This ensures that references from language copies to the shared content is relative. This helps in maintaining language copies and ensures that no patching of references is required when creating or updating language copies.
- If additional attributes of XML tags need to be translated in component-based translation, they can be specified in `/etc/workflow/models/translation/dita_rules.xml`. All additional attribute names need to be prefixed with - `dita_` in the `dita_rules.xml` file. Ensure that you save the ConfigMgr after making this update for the changes to take effect.
- When component-based translation is enabled, references to assets in the source DITA file are automatically changed to point to the same asset under the destination language folder.
- When translating non-DITA assets, it is necessary to make sure that the asset properties that need to be translated are present in `/etc/workflow/models/translation/translation_rules.xml`.
- When translating assets using the translation tab in the DITA map console, it is imperative to accept or reject the translated content using the Accept or Reject buttons in the translation project.

Logging

Custom Logger

- Preferably create a custom logger for `fmdita` specific logs (`com.adobe.fmdita`) so they can be analyzed easily.

- For information on how to create a custom logger, see Logging in AEM documentation.

Open DITA topic or map files in same tab

In some workflows, when you click on a link of a topic or a map file, it opens in a new tab. This could lead to many tabs opened in your browser, which could impact your productivity. You can change this behavior of opening a topic or map file in a new tab, and force it open in the current tab itself. To do so, perform the following configuration changes:

1) Open the Adobe Experience Manager Web Console Configuration page.
   The default URL to access the configuration page is:
   `http://<server name>:<port>/system/console/configMgr`
2) Search for and click on the *com.adobe.fmdita.xmleditor.config.XmlEditorConfig* bundle.
3) Select the **Open DITA Topic/Map in Same Tab** option.
4) Click **Save**.

This setting impacts the following places from where you can access the topic or map files:

- Create DITA Topic (at the end of the workflow, when you click the **Open Topic** button)
- Create DITA Map (at the end of the workflow, when you click the **Open Map** button)
- Topics tab in the DITA map console
- Baselines tab in the DITA map console
- Reports tab in the DITA map console

Unicode support

- To enable support for Unicode characters, open the Configuration Manager and search for *Apache Sling Request Parameter Handling* configuration and set the value of *Default Parameter Encoding* to `UTF-8`.

DTD

- After specifying a Custom DTD in the DITA Profiles, it is imperative to clear the browser cache to make sure that the old DTDs are not used.
- It is recommended that the `catalog.xml` of the Custom DTD contain the Public IDs of all files in the DTD so as to avoid a DTD path resolution error. If this is not possible, it is recommended that the topic at the very least specifies the correct System ID. If the System ID is relative to the local file path from where the file is uploaded, it is recommended to select the **Add System ID Catalog** option at the time of creating or updating the Profile.

Overridden AEM features

- In AEM 6.2, the default inbox and notification features are customized to support the notifications sent during the review workflow. If you make any further changes in the inbox and notification features, you might lose the existing customizations.
- The move, copy & paste, and delete operations for AEM Assets have been overridden. The customized content for the move feature is placed at

`/apps/dam/gui/content/assets/moveassetwizard`, for copy & paste at `/apps/dam/gui/content/assets/jcr:content/actions/primary/pasteasset2`, and for delete feature at `/apps/dam/gui/content/assets/jcr:content/header/items/deletedita` location. If you make any further changes in the move, copy & paste, and delete features, you might lose the existing customizations.

FrameMaker integration

- Users need to have read permission on root (" `/`") node of AEM repository to successfully connect to AEM repository from FrameMaker.
- The file check-in behavior in FrameMaker (2017 release) Update 1 is different from the file check-in experience in the XML web editor. In FrameMaker, when a checked out file is checked back in, the changes are not saved in the latest persisted version, but they are saved in the Latest version that does not have a version number associated with it. This behavior will be changes in the upcoming update for FrameMaker.