

Jorsek Content Development Guide

*easy*DITA

Jorsek LLC
302 Goodman Street North E201
Rochester, NY 14067

Table of Contents

| | |
|--|----|
| Jorsek Content Development Guide..... | 4 |
| Jorsek Style Guide..... | 5 |
| Jorsek Style Guide..... | 5 |
| Abbreviation..... | 5 |
| File Types and Extensions..... | 5 |
| Capitalization..... | 6 |
| Diction..... | 6 |
| Keyboard Key Names..... | 8 |
| Numbers..... | 8 |
| Date and Time..... | 8 |
| Digits..... | 9 |
| Punctuation..... | 9 |
| Bullets..... | 9 |
| Commas..... | 10 |
| Periods..... | 10 |
| Voice..... | 10 |
| Jorsek Information Model..... | 12 |
| Jorsek Information Model..... | 12 |
| Information Types..... | 12 |
| About Topics..... | 13 |
| Concept topic..... | 15 |
| Task topic..... | 16 |
| Reference topic..... | 20 |
| FAQ topic..... | 21 |
| Troubleshooting topic..... | 22 |
| Glossary Entry topics..... | 23 |
| Topic Elements..... | 24 |
| Block Elements..... | 25 |
| Inline Elements..... | 30 |
| Maps..... | 32 |
| Map Structure, Elements, and Attributes..... | 33 |
| Jorsek Strategies..... | 34 |
| Jorsek Strategies..... | 34 |

| | |
|---|----|
| Content Library Structure | 34 |
| Folder Structure..... | 34 |
| Folder Locations..... | 36 |
| Jorsek Content Development..... | 37 |
| Content Creation..... | 37 |
| Workflows..... | 38 |
| File Statuses..... | 39 |
| Linking Strategy..... | 39 |
| Inline Links..... | 40 |
| Reuse Strategy..... | 42 |
| Writing for Reuse..... | 42 |
| Topic and Map Reuse..... | 43 |
| Element Reuse..... | 43 |
| Global Shared Map..... | 45 |
| Metadata Strategy..... | 45 |
| Metadata Guidelines..... | 46 |
| Versioning Strategy..... | 47 |
| Versioning Process..... | 47 |
| Version List..... | 48 |
| Conditional Processing Strategy..... | 49 |
| Conditional Processing Attributes and Values..... | 49 |
| Publishing Strategy..... | 49 |
| DITA Open Toolkit Publishing..... | 50 |
| Jorsek Portal Publishing..... | 50 |
| Quality Assurance Strategy..... | 50 |
| Test Scripts..... | 51 |
| Test Script Locations..... | 51 |
| Creation Process..... | 52 |
| Testing Process..... | 53 |

Jorsek Content Development Guide

With many writers contributing to the content library at Jorsek, it's important to have a well-defined content development guide that all writers can refer to for our Style Guide, Information Model, and Jorsek Strategies. Having this unique content development guide promotes consistency and reduces questions about style, structure, and process.

As the makers of a DITA CCMS, we are especially aware of the benefits that structured authoring provides. We also understand that moving to a structured authoring environment requires a huge shift in processes and strategies. Not only that, but the DITA specification includes hundreds of elements and attributes that are available to use in topics and maps. Having a content development guide is not only critical to creating consistent documentation but also to the effective use of DITA and easyDITA features.

We've made our Content Development Guide public in order to provide our customers, and DITA users in general, with examples of considerations for use in their own content development guides. The guidelines in this guide were made with our content needs in mind. We hope our Guide can help you create a content development guide that fits your organization's needs.

Jorsek Style Guide

Jorsek Style Guide

The Jorsek Style Guide contains guidelines for writing content at Jorsek. With many writers contributing to the content library, having a style guide for our documentation ensures a consistent voice throughout our documentation and reduces questions about style.

We've made our Style Guide public in order to provide our customers with examples of style considerations for use in their own style guide. These guidelines were made with our content needs in mind. You should create a style guide that aligns with your organization's style goals.

Abbreviation

An abbreviation is the shortened form of a word or phrase. Because topics are not always read linearly and any "page" can be the first page, it's important to write out the full term in its first instance.

Guidelines

Keep these guidelines in mind when using abbreviations:

- If the abbreviation is specific to an industry or software and a reader may not recognize it, write it out.
- In every topic, write out the full term, followed by the abbreviation in parentheses. As you continue writing, you can use the abbreviation in place of the full term.

Examples

Each document needs a corresponding Document Type Configuration (DTC). The DTC tells easyDITA what documents to associate with the configuration.

File Types and Extensions

File types and extensions use different style guidelines depending on the specificity of what you are referencing.

Guidelines

Keep these guidelines in mind when using file types and extensions:

- Use all uppercase when referring generally to a file type
- Use lowercase when referring to a specific file

Examples

Upload a PDF.

You should compress all the files in a ZIP.

Include image.png in the folder.

Capitalization

You should limit capitalization to proper nouns (and of course the beginning of sentences and in titles). Always capitalize proper nouns the way they were intended to be capitalized, as dictated by their governing organization.

Guidelines

Keep these guidelines in mind when using capitalization:

- Capitalize names of people, places, organizations, and interfaces.
- Capitalize names the way they were intended to be capitalized, regardless of their location in the sentence. You can start a sentence with a lowercase proper noun.



Note: Capitalization in titles follows specific rules based on the information type. For more information, see *Jorsek Information Model* on page 12.

Examples

easyDITA

oXygen

Dashboard

oXygen enables you to find and replace content.

Diction

Because there are many ways you can express an idea, we've outlined some of our most notable diction choices. You should use the preferred terminology listed here to ensure consistent language throughout our documentation.

General Terms

| Preferred | Nonpreferred | Example |
|------------|------------------------------|---------------------------------------|
| displays | appears | The window displays. |
| enable | allow | The feature enables you to do x. |
| filename | file name | Enter a filename. |
| image | picture, photo | Select an image. |
| screenshot | screen capture, screen image | Upload a screenshot. |
| want | wish, desired | Select the option you want to enable. |

Interface and Components

We maintain our interface names and components in Variable topics. Do not write out interface names. Instead, create a content key reference (`conkeyref`) to the interface name variable in a Variable topic. For more information, see [Variable Topics](#) on page 44.

Screens

| Term | Usage | Example |
|--------|---|-------------------------------------|
| window | large screens with buttons, fields, etc. | In the Bulk Change window... |
| dialog | small screens, usually for errors or confirmations | In the Insert dialog... |
| pane | screens that slide open from the left or right side | In the Activity pane... |

Selectable Areas

| Term | Usage | Example |
|----------------|-------------------------------------|--------------------------------------|
| drop-down menu | selectable list items in a menu | In the More drop-down menu... |
| checkbox | selectable boxes | In the Owner checkbox... |
| field | area where a user types information | In the Keys field... |

Actions

| Term | Usage | Example |
|-------------------|---|---|
| click | interactions with button, icons, or links | In the Content Manager , click the Edit icon. |
| right-click | opening a context menu | In the Content Manager , right-click a file and... |
| select | interactions with a drop-down menu | In the More drop-down menu, select... |
| check and uncheck | interactions with a checkbox | Check the checkboxes next to the files you want to rename. |
| enter | typing text into with a text area | Enter a title for your file. |
| navigate | find a file or folder | Navigate to the image you want to upload. |
| place | location of cursor | Place your cursor where you want to insert a table. |

Keyboard Key Names

We want our content to be as semantically rich as possible. When writing about keyboard names, always use the `Shortcut` element.

Guidelines

Keep these guidelines in mind when using keyboard key names:

- Use the abbreviated form
- Capitalize the first letter of a keyboard key name
- Use the `Shortcut` element nested in the `UI Control` element to write about keyboard keys or the `Shortcut` element nested in the `UI Control` element and `Menu Cascade` element to write about shortcuts requiring multiple steps. For more information, see [Shortcut element](#) on page 31.

Examples

Alt

Cmd

Press Cmd > C to copy.

Numbers

Jorsek uses specific guidelines for using written numbers and digits to ensure readers can easily scan for numeric information.

Date and Time

The date and time should always be written so it's easy to scan for relevant numeric information.

Guidelines

Keep these guidelines in mind when using dates and times:

- Use a 12-hour time format and include the 12-hour period and timezone. The timezone should be abbreviated and in all uppercase.
- Use a lowercase acronym after the numeric time.
- Use a MM-DD-YYYY format for dates or write the date out in full.

Examples

The meeting is at 1:00 pm EST.

The revisions were made on 01-31-2030.

The meeting is on January 31, 2030.

Digits

Use digits for large numbers so it's easier to scan for this information.

Guidelines

Keep these guidelines in mind when using digits:

- Spell out numbers one through nine
- Use numerals for numbers 10+
- Use a comma in numbers of four digits or more

Examples

Select two options.

Select 12 files in the file listing.

There are over 5,000 combinations.

Punctuation

Jorsek follows The Chicago Manual of Style, with some notable punctuation style considerations outlined for reference.

Bullets

Use bullets to make it easier to scan lists of things.

Guidelines

Keep these guidelines in mind when using bullets:

- Use a bullet list when organizing multiple items into a list makes more sense. Typically a list is more effective when there are three or more items in a series.
- Capitalize the first letter of each list item.
- Use parallel structure when writing a bullet list. For example, start each list item with a verb.
- Do not add a period to the end of any list item, even if the list items are complete sentences. The only exception is when a list item includes more than one sentence. In this situation, you must add an end punctuation to all sentences in the list item and all list items in the bullet list.

Examples

The interface enables you to:

- View activity
- Filter events
- Search keywords

Commas

The Oxford comma is the final comma in a series of three or more items. You should use the Oxford comma to reduce ambiguity in your writing.

Guidelines

Keep these guidelines in mind when using commas:

- In a series of three or more items, use a comma before “and” and “or”.

Examples

In the following example, we don't know if the book is dedicated to the writer's parents, Leroy and Joanne, or to the writer's parents, as well as Leroy and Joanne.

- I dedicate this book to my parents, Leroy and Joanne.

To reduce the ambiguity and make it clear that the dedication is to the writer's parents, to Leroy, and to Joanne, we use the Oxford comma:

- I dedicate this book to my parents, Leroy, and Joanne.

Periods

Periods end complete ideas and must be used consistently.

Guidelines

Keep these guidelines in mind when using periods:

- Always use a period at the end of a complete sentence. The only exception to this rule is bullet lists. For more information, see [Bullets](#) on page 9.

Voice

With many authors contributing to our documentation, it's important to use a common underlying tone of voice. In general, we want our content to be simple, objective, and professional. However, we also want to maintain a conversational and engaging tone.

Keep these guidelines in mind when writing:

- Use active voice
- Write in the second person narration
- Keep it simple and direct

If you write following those simple tips, your writing will be clear, precise, and concise. Clear, precise, and concise writing is engaging and effective at transmitting information.

Active voice

In a sentence using active voice, the subject performs the action. In a sentence using passive voice, the subject receives the action, and it can be unclear who is doing the action.

Always try to write in active voice because it conveys the message more clearly and directly. Active voice helps keep the sentence from becoming overly complicated and vague. For example, instead of writing:

- It is recommended that you do not to edit the underlying code.

Write:

- We recommend that you do not edit the underlying code.

Second person narration

We write in second person. Writing in second person addresses the reader directly and engages them in the writing. This means structuring content so that you address the user using "you". For example:

- You can use the History Tab to restore a file to a previous state.

Simple and direct

We want to make our content as easy as possible to read and understand. So, be conservative in adjective and adverb usage, and use simple, common words. For example, instead of writing:

- Utilize the context menu to modify annotations.

Write:

- Use the context menu to modify annotations.

Or instead of:

- Once you have done that...

Write:

- Now...

Jorsek Information Model

Jorsek Information Model

The Jorsek Information Model contains guidelines for structuring content at Jorsek. With many writers contributing to the **Content Library**, having an information model outlining our content structures as well as the elements and attributes used in our documentation is critical to creating consistent documentation.

We've made our Information Model public in order to provide our customers, and DITA users in general, with examples of information architecture considerations for use in their own documentation. These guidelines were made with our content goals in mind. You should create an information model that aligns with your organization's content goals.

Information Types

All content in DITA must be organized into topics based on their information type. We use six DITA information types to organize and structure our content. Each information type contains a specific structure, so choosing the type that best suits the content you're writing and its corresponding topic template is important.

| Information Type | Provides | Topic Template |
|-----------------------|---|--|
| Concept topic | Overview and background information that helps users understand information about an idea, interface, or feature. | Default Concept topic template |
| Task topic | Procedural information that helps users accomplish a goal. | Default Task topic template |
| Reference topic | Reference information, typically organized into tables, that users can refer to when completing a task. | Default Reference topic template |
| FAQ topic | Answers to frequently asked questions asked by users. | Default FAQ topic template |
| Troubleshooting topic | Troubleshooting steps that help users find a solution to software issues. | Default Troubleshooting topic template |
| Glossary Entry topic | Term definitions and terminology information that define terms unfamiliar to new users. | Default Glossary Entry topic template |

About Topics

All content in DITA must be organized into topics, with each topic existing as a standalone chunk of information. Though each topic should live independently of one another, they should all contain some key information that identifies the topic and conveys its relevancy.

Guidelines

Keep these guidelines in mind when creating topics:

- Use the topic template available for the information type
- Assign a title
- Use the Resource ID element
- Use the Short Description element or both the Abstract element and Short Description element

Topic Titles

In the final output, content in the Title element displays as the heading for the topic's content and in the navigation (for example, the table of contents in a PDF or site navigation on a website). Because this content is the first thing an end user sees, it's important that it's clear and concise so they can quickly scan titles to determine their relevance.

Guidelines

Keep these guidelines in mind when writing titles:

- Describe the topic's content
- Use articles (a, an, and the) when necessary
- Avoid using abbreviations

Concept topics

Keep these guidelines in mind when writing Concept topic titles:

- Use title case
- Use nouns or noun phrases

Here are some examples:

Account Summary

Projects Interface

Task topics

Keep these guidelines in mind when writing Task topic titles:

- Use title case
- Begin with a gerund

Here are some examples:

Creating an Assignment

Reassigning a Project

Reference topics

Keep these guidelines in mind when writing Reference topic titles:

- Use title case
- Use nouns or noun phrases
- Include the referenced subject, interface, etc.

Here are some examples:

DITA Open Toolkit Errors

Element Properties

FAQ topics

Keep these guidelines in mind when writing FAQ topic titles:

- Use sentence case
- Write in question format, including ending punctuation

Here are some examples:

What is the maximum upload size?

How do I change the owner of multiple files?

Troubleshooting topics

Keep these guidelines in mind when writing Troubleshooting topic titles:

- Use title case
- Use nouns or noun phrases
- Include the issue or error

Here are some examples:

DITA Open Toolkit Error DX140980

DITA Open Toolkit Error DX123430

Glossary Entry topics

Keep these guidelines in mind when writing Glossary Entry topic titles:

- Use lowercase
- Use the word you're defining

Here are some examples:

branch

release

Short Description element and Abstract element

The Short Description element enables you to provide a short summary, which also acts as an introductory paragraph, for a topic so that end users can quickly determine if a topic is relevant to them. Each topic must have a Short Description element or an Abstract element containing a Short Description element.

Short Description element

Consistent use of short descriptions provides readers with a powerful user experience, enabling them to quickly navigate topics and decide if a topic contains the information they need. They also help clarify a readers understanding of the topic.

Use the Short Description element to provide an introduction to your topics. Because the Short Description element can be used as a link preview of the topic, it shouldn't be longer than two to three sentences. If you need to write a longer introduction to a topic or use elements that are not allowed in the Short Description element—for example, the Note element or Unordered List element—add an Abstract element around the Short Description element.

Guidelines

Keep these guidelines in mind when using the Short Description element:

- Be descriptive
- Do not begin either with "Learn how...", "This topic...", etc.
- Do not restate the title

Here are some examples:

Branching is a versioning tool that enables you to create a parallel set of content in a "branch" so you can edit the branched content without affecting the original content and push changes from one branch to another.

Metadata are tags you apply to your files to classify them. These custom metadata tags must be manually created and assigned to your files, but once they've been created and assigned, you can use them in powerful ways to find and analyze content in your repository.

Abstract element and Short Description element

Whenever you need a more robust introduction to a topic that does not fit into the Short Description element, always combine the Abstract element and the Short Description element. The Abstract element can contain paragraph-level elements that are unavailable in Short Description element.



Note: To insert a <shortdesc> element, you must open the source code and nest the <shortdesc> inside the <abstract> tags.

The example below demonstrates how you can use Abstract element together with the Short Description element:

```
<abstract>
  The abstract provides more complex content.
  <shortdesc>The shortdesc must be directly contained by the abstract.</shortdesc>
  The abstract can put text around the shortdesc.
</abstract>
```

Concept topic

Concept topics should communicate in a clear, concise, precise, and complete way the essential information for understanding a concept. Concept topics answer the question, "What is *x*?"

Keep these guidelines in mind when creating Concept topics:

- Convey the essential information about a concept
- Provide only conceptual information, not reference or procedural information
- Focus only on one particular idea

Some questions to ask yourself when writing Concept topics are:

- Is there relevant background information a reader needs?
- Are there technical notions for understanding *x* that need definition?
- What are *x*'s most important features or characteristics?

- Would an example help clarify what x is?

Concept topic Structure and Elements

Concept topics are not highly structured, with most content contained in the Section element. Use the Default Concept topic template to create a Concept topic, which includes a sample structure and available elements for the information type.

All Concept topics must follow this structure:

- Concept topic
 - Title element (required)
 - Prolog element with Resource ID element (required)
 - Short Description element (required)
 - Conbody element (required)
 - Section element (required, any number)

Elements

We use this primary element in Concept topics:



Note: You can nest any number of block elements and inline elements in these elements.

Section element

Use the Section element to contain subsets of information, typically in the Paragraph element. For more information, see [Section element](#) on page 25.

Concept topic Example: ToasterClassic

The ToasterClassic is the one-of-a-kind, classic machine that's designed to ensure that each slice toasts evenly to your preferred level.

Have perfect slices of delight with your favorite jam or butter. Or if you want to toast something that's frozen, the Defrost feature thaws it before the toaster starts toasting. The ToasterClassic even includes preset settings for:

- Toast
- Waffles
- Bagels
- English Muffin

Task topic

Task topics provide step-by-step procedures for completing a task and answer the question, "How do I...?". Task topics typically contain little or no conceptual information about the task or related idea.

Keep these guidelines in mind when creating Task topics:

- Communicate each step clearly.
- Only include information relevant to completing a step.
- State what the expected result is in the Step Result element for all steps. The last step in a Task topic does not require a Step Result element because the procedure's result is included in the Result element.

- Only include images or screenshots for steps where the interface changes as a result of the step, and it's relevant to illustrate this change to our users. Otherwise, avoid using images and screenshots due to high maintenance costs. Keep these guidelines in mind when inserting images or screenshots:
 - To provide a screenshot of the interface during the step, use the Information element.
 - To provide a screenshot of the interface with completed fields or selections, use the Step Example element.
 - To provide a screenshot of the interface after completing the step, use the Step Result element.



Note: The Step Result element is required as part of our quality assurance. For more information, see [Quality Assurance Strategy](#) on page 50.

Some questions to ask yourself when writing Task topics are:

- What information is required to complete each step in the task?
- What background information does a user need to get started on the task?
- Would images or examples help clarify how to complete the procedure?
- What is the expected result of completing the procedure?

Task topic Structure and Elements

Task topics are highly structured and use task-specific elements, enabling you to clearly document each step as well as any additional information required for a step. Use the Default Task topic template to create a Task topic, which includes a sample structure and available elements for the information type.

All Task topics must follow this structure:



- Task topic
 - Title element (required)
 - Prolog element with Resource ID element (required)
 - Short Description element (required)
 - Taskbody element (required)
 - Prerequisite element (optional)
 - Context element (optional)
 - Steps element (required)
 - Step element (required, any number)
 - Information element (optional)
 - Step Example element (optional)
 - Substep element and Substeps element (optional)
 - Step Result element (required)
 - Result element (required)
 - Postrequisite element (optional)

Elements

We use these elements in Task topics:



Note: You can nest any number of block elements and inline elements in these elements.

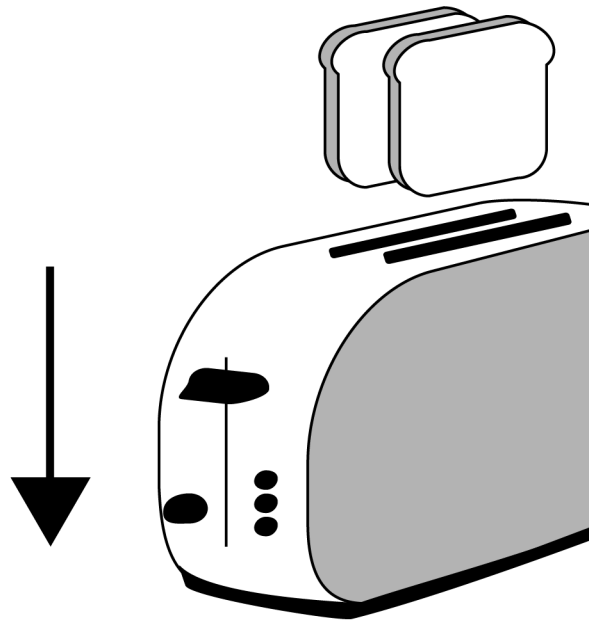
| | |
|------------------------------|--|
| Prerequisite element | Use the Prerequisite element to describe requirements that must be met prior to beginning the procedure. |
| Context element | Use the Context element to provide brief background information for completing the procedure. |
| Steps element | Use the Steps element to contain all steps required to complete the procedure. |
| Step element | Use the Step element to describe an action that must be completed. |
| |  Note: Limit the number of Step elements to 10. Consider creating other Task topics to break a large task into smaller Task topics. |
| Information element | Use the Information element to help clarify a step or provide additional information. Use this element to add screenshots, code blocks, or notes related to a step. |
| Step Example element | Use the Step Example element to provide an example of the action required in the step. Use this element to include a screenshot of how an end user might complete a field. |
| Substeps element | Use the Substeps element to contain all the Substep elements required to complete a step. |
| Substep element | Use the Substep element to break down complex Step elements into substeps. |
| |  Note: Consider creating another task topic if there are many substeps for a step. |
| Step Result element | Use the Step Result element to describe the result of the action performed in the Step element. |
| Result element | Use the Result element to describe the outcome of task completion. |
| Postrequisite element | Use the Postrequisite element to describe requirements an end user must meet after completing the procedure. |

Task topic Example: Toasting Bread

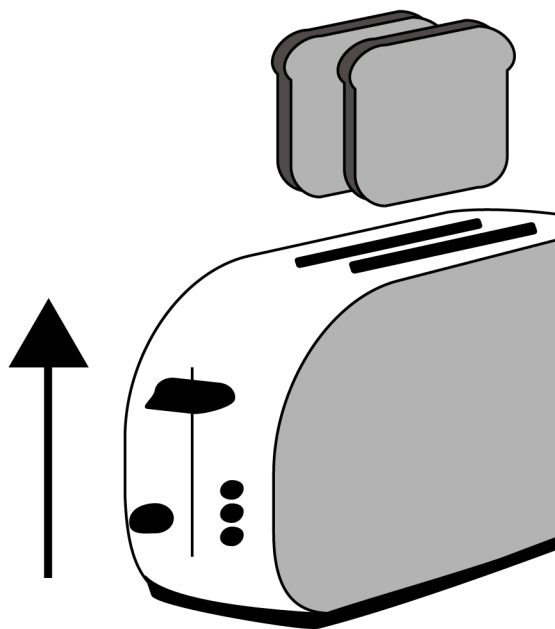
Use the dials and knobs to toast your bread to your preferred level.

You will need:

- A plugged in ToasterClassic
 - Two slices of bread
 - A plate
 - Butter or Jam (optional)
1. Turn the knob to the desired browning level.
 2. Press down the lever to start toasting.



Your bread slices and lever automatically lift up once the toasting cycle is complete.



Your toast is ready for you to spread your favorite jam, butter, or whatever you like on it!

Reference topic

Reference topics provide specification information that supports conceptual information and task completion. These topics provide detailed, quickly accessed data, most often in tables.

Keep these guidelines in mind when creating Reference topics:

- Organize content into scannable structures (tables, bullet lists, code blocks, etc.)
- Provide only reference information, not conceptual or procedural information

Some questions to ask yourself when writing Reference topics are:

- Is there any reference information to support concept or task topics?
- How can I organize this reference content so it can be easily scanned?

Reference topic Structure and Elements

Reference topics use tables that help users quickly scan for information they need. Use the Default Reference topic template to create a Reference topic, which includes a sample structure and available elements for the information type.

All Reference topics must follow this structure:

- Reference topic
 - Title element (required)
 - Prolog element with Resource ID element (required)
 - Short Description element (required)

- Refbody element (required)
 - Section element (required, any number)
 - Table element (optional)

Elements

We use these elements in Reference topics:



Note: You can nest any number of block elements and inline elements in these elements.

Section element

Use the Section element to contain subsets of information, typically in the Table element.

Table element

Use the Table element to create scannable reference material.

Reference topic Example: ToasterClassic Specifications

The ToasterClassic includes many capabilities. The specifications of these capabilities are listed.

| Specification | Description |
|------------------|----------------------------------|
| Capacity (slots) | 2 |
| Output per hour | 30 |
| Loading (kW) | 2.2 |
| Dimensions (cm) | 36 x 21 x 22 |
| Weight (kg) | 4.25 |
| Slot width (mm) | 28 |
| Materials | Stainless steel ends, brass body |

FAQ topic

FAQ topics provide answers to frequently asked questions in a clear and concise question-answer format.

Keep these guidelines in mind when creating FAQ topics:

- Use short, direct titles
- Be concise with your answers

Some questions to ask yourself when writing FAQ topics are:

- Is this a common question or problem?
- Is a short answer sufficient for this question or would it be better answered in the user guide or a tutorial?

FAQ topic Structure and Elements

FAQ topics are simply structured to hold a question and an answer. Use the Default FAQ topic template to create a FAQ topic, which includes a sample structure and available elements for the information type.

All FAQ topics must follow this structure:

- FAQ topic
 - FAQ element (required)
 - Prolog element with Resource ID element (required)
 - Answer element (required)

Elements

We use these elements in FAQ topics:



Note: You can nest any number of block elements and inline elements in these elements.

FAQ element

Use the FAQ element to provide a frequently asked question.

Answer element

Use the Answer element to provide an answer to the question.

FAQ topic Example: Can I change the topic type after creating a topic?

No. Each information type has a unique structure, so you cannot change the topic type of an existing topic.

Troubleshooting topic

Troubleshooting topics describe problems and provide possible remedies. Most often they document common or obscure problems that do not have a single solution.

Keep these guidelines in mind when creating Troubleshooting topics:

- Document common or obscure problems and their remedies
- Include all possible remedies for the problem

Some questions to ask yourself when writing Troubleshooting topics are:

- Is this problem reproducible?
- Do I have all the required information for the situation?
- Is there any additional information required for the remedies?

Troubleshooting topic Structure and Elements

Troubleshooting topics are highly structured to provide clear cause and remedy information. Use the Default Troubleshooting topic template to create a Troubleshooting topic, which includes a sample structure and available elements for the information type.

All Troubleshooting topics must follow this structure:

- Troubleshooting topic
 - Title element (required)
 - Prolog element with Resource ID element (required)
 - Short Description element (required)
 - Troublebody element (required)
 - Condition element (optional)
 - Solution element (required, any number)
 - Cause element (optional, any number)

- Remedy element (required, any number)

Elements

We use these elements in Troubleshooting topics:



Note: You can nest any number of block elements and inline elements in these elements.

Condition element

Use the Condition element to describe the issue or problem that the Troubleshooting topic remedies.

Solution element

Use the Solution element to contain the Cause element and Remedy element.

Cause element

Use the Cause element to describe the source problem.

Remedy element

Use the Remedy element to contain steps that are a potential solution for the problem.

Troubleshooting topic Example: DITA-OT Error DOTX017E

An `<xref>` error using the DITA Open Toolkit.

Publishing using the DITA Open Toolkit.

Found a link or cross reference with an empty `<href>` attribute (`href=""`).

1. Navigate to the file listed with the error.
2. Remove the empty `<href>` attribute or provide a value.

Glossary Entry topics

Glossary Entry topics define a term and its terminology information. Glossary Entry topics are used to build glossaries to ensure a consistent vocabulary.

Keep these guidelines in mind when creating Glossary Entry topics:

- Define one term per Glossary Entry topic
- Define terms concisely

Some questions to ask yourself when writing Glossary Entry topics are:

- Is this a common term?
- Is there relevant terminology information an end user needs?

Glossary Entry Structure and Elements

Glossary Entry topics are highly structured and provide specific elements for defining a term and its terminology information. Use the Default Glossary Entry topic template to create a Glossary Entry topic, which includes a sample structure and available elements for the information type.

All Glossary Entry topics must follow this structure:

- Glossary Entry topic
 - Glossary Term element (required)

- Glossary Definition element (required)
- Glossbody element (required)
 - Glossary Usage element (optional)
 - Glossary Alt element (optional)
 - Glossary Abbreviation element (optional)
 - Glossary Acronym element (optional)
 - Glossary Short Form element (optional)
 - Glossary Synonym element (optional)

Elements

We use these elements in Glossary Entry topics:



Note: You can nest any number of block elements and inline elements in these elements.

| | |
|--------------------------------------|---|
| Glossary Term element | Use the Glossary Term element to provide a term you want to define. |
| Glossary Definition element | Use the Glossary Definition element to provide a definition for a term. |
| Glossary Usage element | Use the Glossary Usage element to provide information about when to use the term. |
| Glossary Alt element | Use the Glossary Alt element to contain alternatives for the term. |
| Glossary Abbreviation element | Use the Glossary Abbreviation element to provide an abbreviation for the term. |
| Glossary Acronym element | Use the Glossary Acronym element to provide an acronym for the term. |
| Glossary Short Form element | Use the Glossary Short Form element to provide a shortened version of the term. |
| Glossary Synonym element | Use the Glossary Synonym element to provide a synonym for the term. |

Glossary Entry Example

An example of a glossary entry topic.

Topic Elements

The elements we use in topics fall into two groups: block elements and inline elements.

Block Elements

Block elements are paragraph-like elements, such as the Section element, that allow for the nesting of inline elements within them. In the final output, block elements typically take up the full width of the output format and result in spacing before and after the element.

Section element

The Section element is an organizational subset of information that is related directly to a single topic. The information in each Section element should be equally important to the topic and so included in the topic.

Keep these guidelines in mind when using the Section element:

- Limit section use to cases where the topic has multiple components that require explanation
- If there are multiple sections in a topic, add a Title element to each section to make scanning the information easier

Sections in Concept topics

Breaking a long Concept topic into sections can make the topic easier to read. This, however, can make the topic more difficult to reuse. You should only break a long Concept topic into sections when the sections can't stand alone without the information in the other sections as context. If the sections can stand alone without the context of the other sections, then consider creating individual concept topics instead.

Paragraph element

The Paragraph element is the most common element used in topics. It is a block of text that contains a single idea.

Keep these guidelines in mind when using the Paragraph element:

- Limit paragraphs to one main idea
- Aim for paragraphs to be between two to five sentences
- Vary sentence lengths
- Avoid transitional words like therefore, first, etc. to improve reusability
- Break up long blocks of text with figures, lists, or other visual elements to help readers easily scan material

Examples

Metadata are tags you apply to your files to classify them. These custom metadata tags must be manually created and assigned to your files, but once they've been created and assigned, you can use them in powerful ways to find and analyze content in your repository.

There are two types of metadata: Taxonomy Metadata and Label Metadata. You use both types of metadata the same way, though they differ in the ability to create new metadata tags on the fly.

Code Block element

The Code Block element contains long portions of code. This element is helpful for providing sample code in steps or as reference material.

Keep these guidelines in mind when using the Code Block element:

- Use the Code Block element to contain two or more lines of code that must be set off from the main flow of text

- Limit the use of the Code Block element to within Task topics and Reference topics
- Indent and space code correctly



Tip: You can add snippets of code within the main flow of text using the Code Phrase element. For more information, see [Code Phrase element](#) on page 30.

Example

```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
```

Definition Entry element

The Definition List element contains a list of terms and definitions.

Keep these guidelines in mind when using the Definition List element:

- Use parallel structure when writing definitions. For example, start each definition with a verb.
- Sort terms alphabetically, unless a strong reason to use some other structure.

Example

Definition

An exact description of a thing or concept.

Term

A word or phrase used to describe a thing or to express a concept.

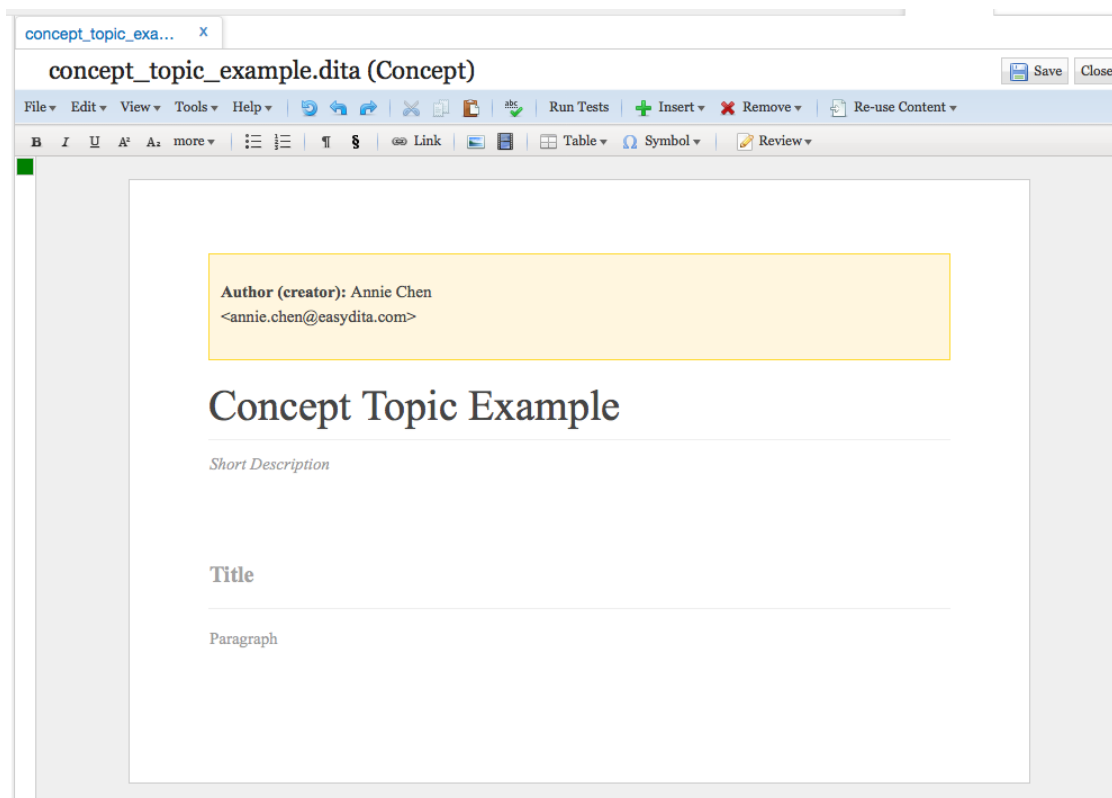
Image element

Images can add value to your content and convey complex ideas. Our publishing plugins handle processing images, but there are some considerations you should keep in mind when using images.

Keep these guidelines in mind when using the Image element:

- Use PNG images so that the images can be used in different screen sizes and for a variety of publishing outputs like HTML, PDF, etc.
- Do not resize images. Let the publishing plugins handle resizing images based on the output format.
- If you're using images in a Task topic to explain a step, we recommend nesting it in another element first. For example, insert an image in the Information element, Step Result element, or Step Example element.
- If you're annotating images, change the annotation color to #FF0000 (red).

Example



List element

Lists enable you to present information in an engaging and easily scannable way. There are two types of lists you can use in your topics: ordered lists and unordered lists.

Guidelines

Keep these guidelines in mind when creating lists:

- Use parallel structure when writing a bullet list. For example, start each bullet point with a verb.
- Capitalize the first letter of each list item.
- Include all common information from the list items in the introductory sentence to avoid repetition.
- Do not use an ordered list to document procedures within a concept or reference topic. Instead, use a task topic.
- Do not add a period to the end of any list item, even if the list item is a complete sentence. The only exception to this rule is when a list item uses more than one sentence. In this situation, you must add an end punctuation to all sentences in the list item and all items in the bullet list must also include end punctuation.

Unordered List element

When you need to list three or more items in no particular order, use the Unordered List element so the reader can quickly scan the bullet list items. Using an Unordered List element prevents long lists of items from being embedded in a sentence, which is more difficult to read.

Ordered List element

The Ordered List element provides some structure so that the list items are numbered. Only use the Ordered List element in a Concept topic to explain a process.



Note: If there is a procedure to follow, use a Task topic instead.

Examples

Modify the following properties:

- Owner
- Status
- Date
- Metadata

Creating and using Taxonomy Metadata requires you to:

1. Create a taxonomy of terms
2. Assign taxonomy to a metadata category
3. Assign metadata to files

Note element

Notes draw an end user's attention to important information. Make sure you use the correct `<note>` type to convey the right message, impact, and relevance.

Guidelines

Keep these guidelines in mind when using the Note element:

- Use the Note element to contain information that is more important than what occurs in the body text
- Add a Note element near, typically directly in or below, the element it relates to

Note Types

We use these common `<note>` types:

| | |
|---------------------|---|
| Default Note | Provides general additional information that needs to be differentiated from the body text. |
| Tip | Provides helpful information that can make a process easier. |
| Important | Provides information that is more important to the user than a general note. |
| Remember | Reminds users to not forget important information. |
| Caution | Warns users when there is potential for a negative outcome. |

Examples



Note: Available options vary based on your configuration.



Tip: To select a range of checkboxes, check the first box in the range and then while holding Shift, select the last box in the range.

Table element

Organize content into tables to create easily scannable content for end users. Our publishing plugins handle processing and styling tables, but there are some considerations you should keep in mind when using tables.

Keeps these guidelines in mind when using the Table element:

- Use the Table element so you can create complex tables with joined cells. Do not use the Simple Table element.
- Use column headings.
- Use title case for all column headings.
- Use consistent grammatical structure for cell content.
- Provide a description of the table's content directly before the Table element itself.

Our publishing plugins handle table styling, so you don't need to worry about:

- Row heights
- Column widths
- Border styling

Example

The following are context menu options:

| Option | Description |
|--------|--|
| Insert | Add an element. |
| Remove | Remove an element. |
| Append | Add an element as a child of the selected element. |

Inline Elements

Inline elements are used in line with the content and can be nested inside block elements. In the final output, inline elements do not take up the full width of the output format and do not result in spacing before and after the element.

Code Phrase element

The Code Phrase element contains a snippet of code that should be used within the main text.

Keep these guidelines in mind when using the Code Phrase element:

- Include snippets of code that are shorter than one line



Tip: You can add long portions of code, set off from the main flow of text, using the Code Block element. For more information, see [code_block_element.dita](#).

Example

The `<template-type=" ">` identifies the template type.

File Path element

The File Path element contains the location of a file, a file name, or a directory.

Keep these guidelines in mind when using the File Path element:

- Include the file extension when referring to a file name
- Use `...` to indicate parent locations that are not required for a user's understanding

Examples

Move `example.png` to a new location.

You can access the files at `.../.../Documentation/User_Guide`.

Menu Cascade element

The Menu Cascade element contains a series of selections related to interface components such as menus or buttons.

Keep these guidelines in mind when using the Menu Cascade element:

- Include series of two or more selections

Example

In the **Topic Editor**, click **Link > Insert Xref**.

Phrase element

The Phrase element contains reusable words or phrases, such as our company name, product name, and interface names.

Keep these guidelines in mind when using the Phrase element:

- Include short pieces of variable content.
- Include commonly used content. This promotes consistency around commonly used words or phrases.



Important: Only create variable content in Variable topics. For more information, see [Variable Topics](#) on page 44.

Shortcut element

The Shortcut element contains keyboard keys and keyboard shortcuts.

Keep these guidelines in mind when using the Shortcut element:

- Follow our guidelines for referring to keyboard keys. For more information, see [Keyboard Key Names](#) on page 8.
- Use the Menu Cascade element to contain keyboard shortcuts containing multiple keyboard keys.

Examples

Press **Cmd**.

Press **Cmd** > **Shift** > **V**.

UI Control element

The UI Control element contains user interface components, such as buttons or menu options. Consistently using this element ensures that all user interface components are styled similarly in the final output.

Keep these guidelines in mind when using the UI Control element:

- Each UI Control element should only contain one user interface component, including:
 - Buttons
 - Menu options
 - Text fields
 - Checkboxes
- Use the Menu Cascade element when referring to a series of components in the user interface. For more information, see [Menu Cascade element](#) on page 30.



Note: Instead of creating a UI Control element in a topic, consider making it an reusable variable so that other writers can use it in their documentation. For more information, see [Variable Topics](#) on page 44.

Examples

Click **OK**.

Uncheck the **Recreate** option.

User Input element

The User Input element contains text that an end user should input into a field, typically a text box.

Keep these guidelines in mind when using the User Input element:

- Limit the use of the User Input element to within Task topics and Reference topics.

Examples

In the Date field, enter 01-31-2018.

Window Title element

The Window Title element contains names of windows, dialogs, or panes.

Keep these guidelines in mind when using the Window Title element:

- Use the Window Title element to contain names of windows, dialogs, or panes.
- Include the interface component type along with the interface name. This means if you're referring to a window, it would be the name of the window plus the word "window".



Important: Instead of creating a Window Title element in a topic, make it a reusable variable so that other writers can use it in their documentation. For more information, see [Variable Topics](#) on page 44.

Examples

In the **Properties window**, select...

In the **Rename dialog**, select...

Xref element

Use the Xref element when you want to link to another topic or external site.

Keep these guidelines in mind when using the Xref element:

- Use the Xref element to add a link to a topic, an element in the same topic, or an external site.



Caution: Do not create cross-references to maps. The DITA Open Toolkit does not gracefully handle links to maps as they are purely organizational structures and generally do not have page or site locations in your final output.

- Do not place links in the middle of a sentence. Instead, place them at the end of the sentence as a supplemental reference.
- Follow our linking guidelines. For more information, see [Inline Links](#) on page 40.

Examples

Follow our guidelines for linking. For more information, see [Inline Links](#) on page 40.

All content must be organized into topics. For more information, see [About Topics](#) on page 13.

Maps

Maps can consist of just topics, have topics nested under other topics, or have submaps nested in them. You can think of maps as binders and topics as pages. To create a deliverable map, you can add individual pages or whole binders. And the way you structure your map dictates the structure in the final output.

Keep these guidelines in mind when creating maps:

- Use the Jorsek Map template to create new maps. This ensures that our reusable content and variables are included in each deliverable.
- Use general, though clear, titles for maps.
- Consider adding a Navtitle element for a submap so a title displays for the map in the output navigation (for example, the table of contents or site navigation).
- Consider creating submaps to organize content that relates to a single, main topic so that the submap can be reused in other maps.

Map Structure, Elements, and Attributes

Maps are not highly structured, which enables you to easily create hierarchy. Use the Jorsek Map template to create a Jorsek Map, which contains our shared map that resolves references like variables or content references (<conrefs>).

All maps must follow this structure:

- Jorsek Map
 - Title element (required)
 - Topicref element (optional, any number)
 - Mapref element (optional, any number)
 - Topichead element, Topicmeta element, and Navtitle element (optional, any number)
 - Mapref element (optional, any number)
 - Reltable element (optional)

Elements

We use these elements in the Jorsek Map:

Topicref element

Use the Topicref element to add a topic to a map.

Mapref element

Use the Mapref element to add a map to a map.

Topichead element, Topicmeta element, and Navtitle element

Use the Topichead element, Topicmeta element, and Navtitle element structure to add a navigational heading for a map.

Reltable element

Use the Reltable element to add a relationship table to a map. This is how you build a related links area to your topics in your final output.

Attributes

We use these attributes in the Jorsek Map:

Resource-only Processing Role

Apply the Resource-only attribute value to a resource so it does not display in the final output, but is used to resolve references. This attribute is typically used on Variable topics and Warehouse topics.

Jorsek Strategies

Jorsek Strategies

As our **Content Library** continues to grow, having well-defined processes and strategies for organizing and managing our content is critical. This makes it easier for us to find, use, and analyze our content.

We've made our content strategies public in order to provide our customers, and DITA users in general, with examples of strategy considerations for use in their own documentation. These strategies were made with our content needs in mind. You should create strategies that aligns with your organization's content goals.

Content Library Structure

The **Content Library** is where our files are stored and accessed. It's comprised of folders, which contain other folders, topics, or maps. Because there are a number of ways you can create and nest folders to organize your content, we've developed a strategy for organizing our **Content Library** in a consistent and logical way. This makes it easier for team members to manage, find, and reuse content.

Our **Content Library** consists of several repositories. For the purposes of this strategy, we'll look at our Documentation repository, which contains all of our documentation. This folder is located at `Library/Documentation`.

Folder Structure

Folders are the primary organizational structure used to sort similar files in our **Content Library**. They can contain subfolders to help break down large content sets into smaller batches. These 'main' folders and their subfolders can also contain common folders with reusable files. This strategy enhances the user experience moving through the folder structure and promotes findability.

You should always create a main folder for your content in one of the existing folder locations in the Documentation folder. And then create subfolders for areas or categories within the main folder. For more information about existing folder locations where you can create folders, see [Folder Locations](#) on page 36.

Here's an example of a new folder structure in `Library/Documentation/User_Guide`:

- Documentation
 - User_Guide
 - Main folder
 - _Media folder
 - _Global_Shared folder
 - Subfolder
 - _Media folder
 - topicA.dita
 - topicB.dita
 - Subfolder
 - _Media folder

- topicC.dita
 - topicD.dita
- example.ditamap
- Main folder
 - _Media folder
 - _Global_Shared folder
 - Subfolder
 - _Media folder
 - topicE.dita
 - topicF.dita
 - Subfolder
 - _Media folder
 - topicG.dita
- example.ditamap

Common Folders

Within each logical place in your folder structure, you should also create common folders to hold reusable files. For more information about reusable content, see [Element Reuse](#) on page 43.

We organize our reusable files into three types of common folders based reach and type of reusable content:

- Global Shared Folders
- Shared Folders
- Media Folders

Global Shared Folders

Global content is content that is used in every publication. We have a Global Shared folder at the highest level at `Library/Documentation/_Global_Shared` that contains reusable content used throughout all deliverables. This makes it easy to find, use, and edit our global reusable files.

Within this folder, we have:

- DITA Vals
- Glossary Terms
- Media
- Variables
- Warehouses

Keep these guidelines in mind when creating Global Shared folders:

- Prefix Global Shared folders with an underscore to ensure they float to the top of the file listing. For example, `_Global_Shared`.
- Create a Global Shared folder at the highest level in the folder structure so that it's easy to find.

Shared folder

Shared content is content that is used in topics or maps within the same main or subfolder. We have Shared folders in our main folders and sub folders when there is reusable content used only within those folder structures.

Keep these guidelines in mind when creating Shared folders:

- Prefix Shared folders with an underscore to ensure they float to the top of the file listing. For example, `_Shared`.
- Do not create a Shared folder if there are no reusable files at that level.

Media Folders

Media folders contain media assets, including images and videos. We organize our media files into media folders at the highest level so that they're easier to find.

Keep these guidelines in mind when creating Media folders:

- Prefix Media folders with an underscore to ensure they float to the top of the file listing. For example, `_Media`.
- Do not create a Media folder if there are no media assets at that level.

Folder Locations

All documentation files must be organized into one of the following folders in the Documentation repository.

| Folder | Contains |
|---|--|
| Jorsek Content Development Guide | Jorsek's Style Guide, Information Model, and Strategies. |
| Global Shared | Files that support reuse, such as Warehouse topics and Variable topics. |
| Personal Spaces | A personal folder for each Jorseker so they can create test files. |
| Client SDK | Content about the client software development kit. |
| Configurations and Customizations | Guides for available easyDITA customizations and configurations. |
| Connectors, Plugins, and Third-Party Integrations | Guides about our connectors, plugins, and third-party integrations. |
| Customer Customizations | Client-specific customizations. |
| Documentation Portal | Our customer-facing deliverable map with easyDITA documentation. |
| easyDITA Portal Development | Guide for developers to set up customer easyDITA portals. |
| InfoDev Staging | Centralized staging folder for all in progress InfoDev projects. |
| Instructive, Learning, and Training | Learning materials for using easyDITA and DITA in general, including <ul style="list-style-type: none"> • FAQs • How-To's • Tutorials • Strategies |

| Folder | Contains |
|-------------------------|--|
| Internal Knowledge Base | Reference information for Jorsek employees, such as guides, company policies, and development documentation. |
| Release Notes | Release notes for each version of easyDITA. |
| Tests | Customer Success Developer checklist for programming tests. |
| User Guide | Current easyDITA user guide documentation. |
| xquery | Documentation on the available functions when writing xqueries. |
| zzzUnorganized | A grouping of incomplete and/or random files yet to be incorporated into our documentation. |

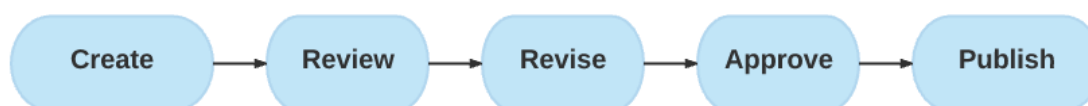
Jorsek Content Development

Our content development process follows a simple path from creation and review to publish. This process means that all content moves through the same general workflow and allows us to create clear, comprehensive, and accurate content for our end users.

The content development process begins with you creating content and organizing the content into a map. Once your map is ready to be reviewed and approved, you can create an assignment for the map using one of our workflows. For more information about our workflows and review process, see [Workflows](#) on page 38.

Once the content has been reviewed, it can be published to multiple formats. For more information about our publishing process, see [Publishing Strategy](#) on page 49.

Figure 1: General Content Development



Though this general content development diagram is illustrated linearly, it's not uncommon to move backwards in the workflow to repeat some steps again. This happens most often when a topic has been reviewed and revised but the revisions were substantial enough to require another review.


Content Creation

When it comes to writing in easyDITA, we recommend thinking with a map-centric approach. Maps not only enable you to create deliverables by referencing individual topics and other maps, but they also resolve references.

Always start a writing project by creating a map and opening the map in the **Map Editor**. Leave the map open in the **Map Editor** as you create new topics, write content, and add topics to the map. Doing this

ensures that the map that you have open is used as the context map to resolve references such as variables. For more information, see [Variable Topics](#) on page 44.

Without a map open in the **Map Editor**, the variables used in our documentation will not resolve and cannot be selected for use. This means that when viewing or editing topics containing variables, you'll see the variable path instead of the variable text. This will make editing and viewing content difficult.


 **Important:** You must create maps using the Jorsek Map template. For more information, see [Map Structure, Elements, and Attributes](#) on page 33.

Without a map open in the **Map Editor**, you would see the following when editing a topic in the **Topic Editor**:

`varsProperNouns/productName` enables you to create branches and manage versioned content using the `vars/branchingT`.


With a map open in the **Map Editor**, you would see the following when editing a topic in the **Topic Editor**:

`easyDITA` enables you to create branches and manage versioned content using the **Branching Tab**.

 **Note:** You can manually set a context map via the **Context Map** drop-down menu to resolve dependencies such as variables. If you open a map after manually setting a context map, the manually set context map will override the context in the open map.

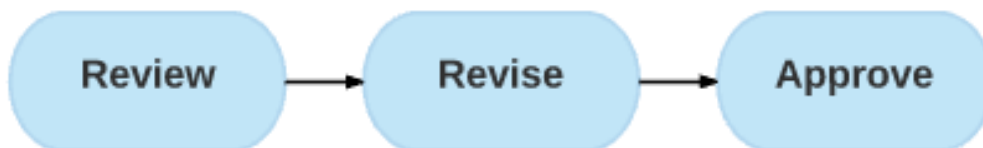
Workflows

Once you're done writing your content, you need to create an assignment for the content to ensure it is reviewed and approved prior to publication. We have two customized workflows, either of which all our content must move through to ensure it is accurate and consistent with our style guide, information model, and strategies. These workflows vary in the amount and type of review required, but both contain review, revise, and approve steps.

 **Note:** Our workflows generally move linearly, though you can move steps backwards to repeat steps. For example, when substantial changes made during a revision require another review.

Simple Review

We use the Simple Review workflow for quick reviews of content that typically require only one review prior to approval.



After authors develop content:

1. A reviewer reads the material and suggests changes.
2. The author makes appropriate revisions.
3. The final reviewer reads the material for approval.

Technical and Editorial Review

We use the Editorial and Technical Review workflow for content that requires a strict review cycle of two reviews, one to ensure the content's technical accuracy and another to make sure the content aligns with the style guide.



After authors develop content:

1. A technical reviewer reads the material and suggests changes.
2. The author makes appropriate revisions.
3. An editorial reviewer reads the material and suggests changes.
4. The author makes further revisions.
5. The final reviewer reads the material for approval.

File Statuses

As content moves through our assignment workflows, it is automatically or manually assigned file statuses based on its stage in the content development process.

We use these file statuses:

| | |
|---------------------------|---|
| Obsolete | File is no longer needed in any topic or map. |
| Needs Reevaluation | File is out of date, incorrect, or incomplete and needs to be reevaluated. |
| In Progress | File is currently being worked on. |
| In Review | File is being reviewed and in an assignment workflow. |
| Approved | File has been reviewed and approved. |
| Ready to Merge | File has been approved and needs to be merged with the most recent branch. For more information, see Versioning Strategy on page 47. |



Note: Files set to the Obsolete or Needs Reevaluation will be reviewed and possibly moved to the trash during a content audit.

Linking Strategy

Adding links to your topics can help direct our end users to related, supplementary information. These links, however, create dependencies between the topic containing the link and the topic or external resource being linked to. Follow our linking strategy to ensure that your links are created consistently and are easy to identify and maintain.

Depending on where you want the link to display, you'll use either inline links or relationship tables.

Inline Links

An inline link displays inline with your content. This means that you can use the text around the link to provide context and explain the link's relevancy. Our inline links are created by inserting cross-references (<xrefs>) to topics, elements (local links), or external resources (websites).



Caution: Do not create cross-references to maps. The DITA Open Toolkit does not gracefully handle links to maps as they are purely organizational structures and generally do not have page or site locations in your final output.

Link to Topics

Creating an inline link to a topic enables you to direct users to related topics with supplementary information. We create inline links by inserting a cross-reference (<xref>) in a topic that references another topic.

Guidelines

Keep these guidelines in mind when creating <xrefs> to topics:

- Do not manually set link text. By default the publishing engine will use the topic's title as the link text. So if the topic's title changes later on, the link text will update to use the new title.
- Do not use the link text as part of a sentence. For example, "We offer [guides] and training." If the guides topic is later renamed to something else, the text surrounding the link may not make sense anymore.
- Structure <xref> links as "For more information, see [<xref>]." to ensure consistent link structures throughout our documentation. You can use this sentence directly after information it's related to.
- Inline links to topics should be used sparingly, as this creates a link dependency between the two topics that requires both the topic with the link and the topic being linked to be in the map for the link in the topic to resolve.

Examples

Here's an example of an <xref> to a topic:

Here is a sentence about links. Here is another sentence about links. Here is another sentence about links. Here is yet another sentence about links. For more information, see [<xref> to topicA.dita].

When published, if the topic title is "Topic A" then the <xref> link text would render as:

Here is a sentence about links. Here is another sentence about links. Here is another sentence about links. Here is yet another sentence about links. For more information, see [Topic A].



Note: Content in the square brackets indicate hyperlink text.

Link to Elements

Creating an inline link to a topic enables you to direct users to specific content in a topic. We create inline links by inserting a cross-reference (<xref>) in a topic that references an element ID. This type of link is also referred to as a local link and is most commonly used to refer to steps or sections in the same topic.

Guidelines

Keep these guidelines in mind when creating <xrefs> to elements:

- Only create local links to elements in the same topic. This limits the link dependencies to within the same topic.

- Do not manually set link text. By default the publishing engine will use the element's link text (if applicable) or element number (for Step elements) as the link text. So if the element's link text or element number changes later on, the link text will update to use the new text.
- Assign element IDs that include the element type and a short summary. For example, an element ID on a step that describes how to edit the assignment summary text would be `stepEditAssignmentSummaryText`.

<xref> to a Section element

Here's an example of an <xref> to a section:

For more information, see [[<xref> to a section with the element ID `sectionAssignmentSummary`](#)].

When published, if the section title is "Assignment Summary" then the xref link text would render as:

For more information, see [Assignment Summary].



Note: Content in the square brackets indicate hyperlink text.

<xref> to a Step element

Here's an example of an <xref> to a step:

Repeat steps [[<xref> to step with the element ID `stepEditAssignmentSummaryText`](#)] to [[<xref> to step with the element ID `stepCloseAssignmentSummary`](#)].

When published, if the first <xref> is in the second Step element in the Task topic and the second <xref> is in the fourth Step element, then the <xref> link text would render as:

Repeat steps [2] to [4].



Note: Content in the square brackets indicate hyperlink text.

Links to Websites

Creating an inline link to an external resource, such as a website, enables you to direct users to supplementary content. We create inline links by inserting a cross-reference (<xref>) in a topic that references an external resource.

Guidelines

Keep these guidelines in mind when creating <xrefs> to websites:

- Set the **Source (href)** field to `external` so our publishing engines know how to handle the link.
- Set the link text. Because the content being linked to is outside of easyDITA, easyDITA is unable to generate link text.
- Add links to external resources to our warehouse file so we can easily manage the links as these resources are updated and links change. For more information, see [Warehouse Topics](#) on page 43.

Examples

Here's an example of an <xref> to a website:

The XYZ site provides information about their course offerings and pricing. For more information, see [[<xref> to `www.example.com`](#)].

When published, if you set the link text to "Example.com" then the `<xref>` link text would render as:

The XYZ site provides information about their course offerings and pricing. For more information, see [Example.com].



Note: Content in the square brackets indicate hyperlink text.

Reuse Strategy

You can reuse entire topics or maps, or elements in a topic. Reusing topic and maps is pretty straightforward, while reusing elements requires centralizing these reusable elements. Our reuse strategy is designed to make finding candidates for reuse and updating reusable content easy by storing reusable content in a central location.

Reusing content enables us to write our content once and reuse it several times. When you reuse content, you're not copying and pasting information. Instead you're creating a reference to the source content. This means that when you make a change to the source content, the change is reflected in every instance it's reused. Reusing content also promotes consistency throughout our documentation since ideas aren't conveyed differently.

Writing for Reuse

You should write content with reuse in mind. Learning to write for reuse requires you to move away from writing in the traditional, linear writing style and towards modular writing.

To do this, avoid restricting your writing with the following phrasings:

| Phrasing | Explanation | Examples |
|----------------|---|--|
| Transitions | Content with transitional phrases can seem out of place when reused in another context, especially when they're not used in the same order. | First, as discussed earlier, etc. |
| Positions | Content with positions listed are difficult to reuse because the object's location can change in a different interface. Listing a position also makes it difficult to maintain should the position ever change. | Click Properties on the right. |
| Demonstratives | Content referring to ideas as "this" or "it" can't be reused. Demonstratives require context from surrounding paragraphs to understand what "this" or "it" refers to. | It typically works best when you refresh the browser first. If this doesn't work, then close the window. |

Topic and Map Reuse

Reusing entire topics or maps in other maps is a very common and simple form of reuse. This form of reuse is convenient for reusing standard content in multiple publication maps.

Keep these guidelines in mind when reusing topics and maps:

- Do not reuse a topic or map multiple times in the same map. Consider using conditional processing instead so the topic only needs to be used once in the deliverable.
- Move reusable topics to a common folder. For more information, see [Folder Structure](#) on page 34.

Element Reuse

You can reuse topic elements by creating a content reference (<conref>) in a topic to the element you want to reuse. Because element reuse is done on a granular level, these reusable elements must be stored in a central location such as a Warehouse topic.

Keep these guidelines in mind when reusing elements:

- Do not reuse content in topics that are not Warehouse topics. This creates dependencies that are difficult to manage.
- If an element can be reused in multiple topics, create it in a Warehouse topic and then create a conref to that reusable element in the Warehouse topic. For more information, see [Warehouse Topics](#) on page 43.
- If an element is reused in multiple topics, move the element to a Warehouse topic and edit the existing <conrefs> to reference the element in its new location instead.
- Ensure that your cursor is in the correct location when inserting a <conref>. Element reuse is contextual, so you can only reuse elements that are valid based on your cursor location.

Warehouse Topics

Warehouse topics are generic topics that are used solely to store reusable elements. We use Warehouse topics to store reusable elements in a central location. You should always create reusable content in a Warehouse topic so that it's easy for others to find the reusable content to create a content reference (<conref>) to it.

Keep these guidelines in mind when using Warehouse topics:


- Whenever possible, create reusable content in an existing Warehouse topics instead of creating a new Warehouse topic.

Keep these guidelines in mind when creating Warehouse topics:

- Create new Warehouse topics in `Documentation/_Global_Shared/Warehouses`.
- Use the Default Topic template to hold all reusable elements, except for reusable task elements. Instead use the Default Task topic template to hold reusable task elements.
- Title the Warehouse topic with the category for the warehouse and include "Warehouse". For example "Map Editor Steps Warehouse".
- Add the Warehouse topic to our Global_Shared map. For more information, see [Global Shared Map](#) on page 45.

Warehouses Topics

We organize our reusable content into several warehouse topics based on the area of the software or element type.

| File | Contains |
|-------------------------------------|---|
| <i>Configurations Warehouse</i> | Reusable topic elements required for configurations. |
| <i>Localization Steps Warehouse</i> | Reusable task elements required for documenting the Localization Manager . |
| <i>Map Editor Steps Warehouse</i> | Reusable task elements required for documenting the Map Editor . |
| <i>Notes Warehouse</i> | Reusable note elements. |
| <i>oXygen Steps Warehouse</i> | Reusable task elements required for documenting oXygen. |
| <i>Properties Warehouse</i> | Reusable elements for documenting properties. |
| <i>Publishing Steps Warehouse</i> | Reusable task elements for documenting the Publishing interface. |
| <i>Steps Warehouse</i> | Reusable steps.  Note: This Warehouse topic has been deprecated in favor of the dedicated warehouse topics for each software area. |
| <i>Topic Editor Steps Warehouse</i> | Reusable task elements for documenting the Topic Editor . |
| <i>WordPress Steps Warehouse</i> | Reusable task elements for documenting WordPress. |

Variable Topics

Variables are reusable phrases, most commonly used for our interface, product, and company terms. Instead of writing out this content, always use a variable in its place. All variables must be created in a Variable topic. That way, when we need to make changes to a variable, they're in a central location.

Keep these guidelines in mind when creating variables and Variable topics:

- Whenever possible, add a variable to one of the existing Variable topic instead of creating a new Variable topic.
- Create new Variable topics in `Documentation/_Global_Shared/Variables`.
- Use the Phrase element to contain variable text.
- Assign element IDs that easily and broadly identify the variable text.

Variable Topics

We organize our variables into several Variable topics based on the software area they relate to.

| File | Contains variables for |
|-------------------------------------|---|
| <i>DITA Elements and Attributes</i> | DITA-specific elements and attributes |
| <i>Interface Variables</i> | Main easyDITA interface names |
| <i>Interface Components</i> | Components of easyDITA's interfaces, such as windows, dialogs, etc. |
| <i>Link Variables</i> | Links to external sites |

| File | Contains variables for |
|------------------------------|---|
| <i>MindTouch Variables</i> | MindTouch specific terminology |
| <i>Proper Noun Variables</i> | Product and company proper nouns |
| <i>Reports Variables</i> | Report names |
| <i>Sales Variables</i> | Sales-related terminology for our modules and subscriptions |

Global Shared Map

We use a Global Shared map to house all of our reusable content, which is then used to resolve references such as variables and content references (<conrefs>). Organizing this content in one map makes it easier to make changes in one location and have changes reflected everywhere the map is reused.

Keep these guidelines in mind when using our Global Shared map:

- The Global Shared map is stored at `Library/Documentation/_Global_Shared`
- Each deliverable map must contain our Global Shared map
- The Global Shared map must have the resource-only processing-role attribute



Tip: If you use the Jorsek Map template to create a new map, then your map already includes a map reference (<mapref>) to our Global Shared map set as a resource-only reference.

Metadata Strategy

As our **Content Library** grows, the ability to find content quickly becomes critical to reusing content as well as inventorying and auditing our **Content Library**. Tagging our content with metadata enables us to do this.

easyDITA automatically tracks CMS-level metadata associated with a file, including:

- Last Modified Date
- Created Date
- Validation
- Broken Links
- Open Comments
- Owner
- Locked by
- Status
- Content Type

This CMS-level metadata enables us to determine important information such as if files are due for a rewrite, how many files are in review, in progress, or approved, and whether we have enough Concept topics to support our Task topics.

In addition to CMS-level metadata, we have custom metadata. This custom metadata is built on a taxonomy structure of terms with hierarchical relationships. Our custom metadata can be manually applied to files individually or by bulk.

This custom metadata enables us to classify our content into metadata categories and gather more information about our **Content Library**.

Metadata Guidelines

In easyDITA, we use custom metadata to classify and audit our content. Whenever you create a topic or map that is customer-facing, you must assign metadata to it.

Keep these guidelines in mind when assigning custom metadata:

- Use the preconfigured custom metadata tags
- Apply metadata on all customer-facing files, including topics, maps, and media assets
- Whenever possible, assign metadata at creation in the **Create New** window
- When applying after file creation, do this by bulk on a map and its dependencies
- Assign the metadata tag that directly applies to the content
- Each file must be assigned at least one metadata label

Metadata Categories

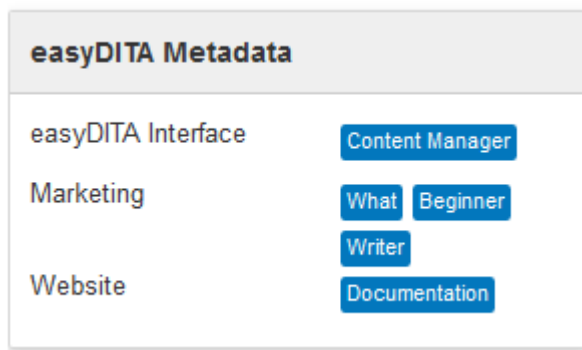
Our custom metadata is sorted into the following categories:

| Metadata | Description | Use to |
|-----------------------------|---|--|
| easyDITA Interface Metadata | Areas of the easyDITA interface | Identify content about easyDITA interfaces |
| Marketing Metadata | Criteria of marketing content | Identify content complexity and intended persona |
| Rebranding Metadata | Criteria for rebranding | Identify content requiring updates for rebranding efforts |
| Internal QA Metadata | Fail or pass information for internal quality assurance | Identify task topics that have passed or failed internal quality assurance testing |
| Content Defect Metadata | Information gap or outdated data | Identify content that contains information gaps or is outdated |

How many metadata labels should I use?

Each file must be assigned at least one metadata label from one of the categories. In most cases, you'll probably assign more than one metadata tag to a file.

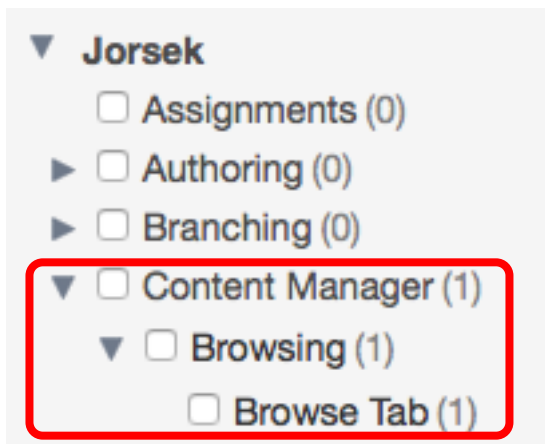
For example, the Content Manager Overview topic in the user guide is assigned the following metadata tags:



How do I know which metadata tags to use?

When assigning metadata, you should only assign the tag that directly applies to the content.

For example, I created the Browse Tab topic for the user guide. I will only assign the Browse Tab metadata to the file, even though the Content Manager and Browsing tags both apply:



The Browse Tab topic is only assigned the Browse Tab metadata tag because it is the child of the Browsing and Content Manager tags. So, the Browse Tab topic will still display when filtering with the parent labels. Think of Browse Tab as inheriting its parent labels. So, when you assign metadata to a file, you don't need to assign all related metadata tags. Just use the tag that directly applies to the content.

Versioning Strategy

With several releases of our software coming out each year, it's important that we have a well-defined process for creating and maintaining different versions of our content. Our versioning strategy uses two versioning methods: releasing and branching. This ensures that we have an archived copy to fallback on and an editable version that we can revise and push changes to.

Versioning Process

Our versioning process uses both releases and branches and is focused on versioning content in the Master branch. The Master branch contains our most up to date content, including customer-facing and internal documentation.

Keep these guidelines in mind when versioning:

- Our customer-facing documentation is organized into our Portal Map located at `Library/Documentation/Documentation_Portal`. This is the only map that gets versioned.
- We use statuses to identify content ready to be versioned or merged. For more information, see [File Statuses](#) on page 39.
- Content in the Master branch is the most current version of our documentation.
- Our customer facing documentation versioning is aligned with our software versioning.

Versions

The Master branch contains our most up to date content and is only versioned when the Portal Map and its dependencies are approved. Versioning this content requires creating a release and a branch for the map at its

current state. The versioned content then exists in a branch where it can be modified while the Master branch continues to exist as the next version of the content. You should always make changes in the Master branch.



Caution: Each new branch should be created from the Master branch, this ensures that a connection between the Master branch and the new branch is created and so you can easily merge new changes in the Master branch to the new branch, or vice versa.

Keep this process in mind when creating a new version of content:

1. Ensure that all customer-facing topics and maps are set to Approved.
2. Ensure that all customer-facing topics and maps are included in the Portal map.
3. In the Master branch, create a release for the Portal Map called “Release #”. For example, “Release 17.0”.
4. In the Master branch, create a branch for the Portal Map called “#”. For example, “17.0”.

By creating a release we have an archived copy of our versioned content at its current state. This is a good fail-safe in case we ever need to return to this state. By creating a branch, we have an editable version of the content in a separate space.

Updates

For the most part, versioned content in branches is rarely revised. It’s more likely that if content needs to be revised, then the current docs probably need the update, too. So you should update the content in the Master branch and then merge the change from the topic or map in the Master branch to the topic or map in the target branch. For example, you may want to merge changes you make in the Editing Assignments topic in the Master branch with the Editing Assignments topic in the 17.0 branch to ensure that the documentation in the 17.0 branch is up to date since the changes also apply to the 17.0 version of the content.

Keep these guidelines in mind when pushing changes from one branch to another:

- Ensure that all customer-facing topics and maps are set to Ready to Merge
- Whenever possible, use the Master branch as the source branch

Version List

We version our customer-facing documentation with each new software release.

The following versioned content is in the Portal Map located at `Library/Documentation/Documentation_Portal`.



Note: We create and stored all releases in the Master branch.

| Software Version | Release Name | Branch Name | Branch Location |
|------------------|--------------------|-------------|---|
| 16.1 | N/A | 16.1 | 16.1/ Documentation/ Documentation_Portal |
| 16.2 | 16.2 Documentation | 16.2 | 16.2/ Documentation/ Documentation_Portal |
| 17.0 | 17.0 Release | 17.0 | 17.0/ Documentation/ Documentation_Portal |

Conditional Processing Strategy

Within our topics and maps, we include audience-specific content for particular user groups. We use DITA's Conditional Processing Attributes to 'tag' topic or map elements as audience-specific. Then at publish, our publishing engine can process content with these tags differently to include or exclude the content based on our publishing parameters.

Keep these guidelines in mind when using Conditional Processing Attributes:

- Conditional Processing Attribute values are case sensitive. You must use consistently apply the values already in use. For more information, see [Conditional Processing Attributes and Values](#) on page 49.
- You can apply Conditional Processing Attributes to topic elements or map elements.
- Separate multiple values with a space to use multiple Conditional Processing Attributes on an element.
- When applying Conditional Processing Attributes in topics, assign values at the nearest parent element to ensure the element is correctly conditionalized. For example, in a bullet list, conditionalizing just the Paragraph element means, when the value is excluded, only the Paragraph element will be excluded, leaving an empty List Item element. Instead you should assign the value to the List Item element to exclude that list item.

Conditional Processing Attributes and Values

Conditional Processing Attributes enable us to conditionalize our content so that it can be processed differently at publish to include or exclude the conditionalized content.

Here are the Conditional Processing Attributes and values that we set on our content:

| Conditional Processing Attribute | Value | Content Meant for |
|----------------------------------|-------------|---|
| audience | internal | Internal audiences |
| audience | userTesting | Testers performing internal quality assurance on the software and documentation |
| audience | all_dev | All developers |
| audience | core_dev | Core Team |
| audience | csd | Customer Success Developers |
| audience | csm | Customer Success Managers |
| audience | marketing | Marketers |
| audience | sys_admin | System Administrators |

Publishing Strategy

Our DITA content is highly structured and can be outputted to multiple formats. To output our DITA content to a format accessible to our end users, we need to publish it. Because our content contains conditionalized

content, it's important that we use the correct publishing scenario and parameters to ensure customer-facing, branded deliverables are created.

DITA Open Toolkit Publishing

You can use the DITA Open Toolkit to transform your content to various output formats. We typically only use the DITA Open Toolkit when publishing to PDF.

Keep these guidelines in mind when using the DITA Open Toolkit to publish:

- Ensure that the files have been reviewed and approved (set to the Approved file status) before publishing
- For PDF output:
 - Use the Jorsek PDF publishing scenario
 - Select the jorsek.pdf transform
 - Use our ditaval file `Documentation/_Global_Shared/customer_facing.ditaval` as the args.filter to ensure internal information in our documentation is removed at publish

Jorsek Portal Publishing

Our Jorsek Portal enables us to display our documentation as web content, making it accessible to our end users. The Jorsek Portal includes our customer-facing documentation, including our User Guide, Tutorials, and Jorsek Strategy docs.

The Jorsek Portal is hardcoded to automatically publish content contained in specific maps. Each map that is hardcoded to the Jorsek Portal contains our versioned documentation in its corresponding branch.

The following table lists the versioned documentation maps available and the corresponding branch and location of these maps:

| Version | Branch | Location |
|-----------------|--------|---|
| Current content | master | https://jorsek.easydita.com/share/d7769f00-ce5b-11e6-b664-42010a8e0005 |
| 17.0 | 17.0 | https://jorsek.easydita.com/share/3f741dd0-fa9e-11e6-8095-42010a8e0005 |
| 16.2 | 16.2 | https://jorsek.easydita.com/share/6451cae0-db99-11e6-b664-42010a8e0005 |
| 16.1 | 16.1 | https://jorsek.easydita.com/share/5df4e880-37f7-11e6-8fbf-bc764e105744 |

Quality Assurance Strategy

Before new versions of easyDITA are released, we test the release version internally using test scripts in our User Guide. This enables us to discover and fix bugs in the software and issues with the documentation

before the new version is released. Our quality assurance strategy involves two processes: creating test scripts and testing test scripts.

Test Scripts

Test scripts are Task topics that contain procedures for testing a feature or bug fix. Our testing only requires that Task topics in the User Guide are tested.

There are two types of test scripts:

User Guide Test Scripts

User Guide test scripts are Task topics that are included in the User Guide and double as customer-facing documentation. We're able to use these topics in both our User Guide and for quality assurance by conditionalizing content in the topics that's only meant for quality assurance purposes. For more information, see [Conditional Processing Strategy](#) on page 49.

Dedicated Test Scripts

Dedicated test scripts are Task topics that are not included in the User Guide and exist exclusively for quality assurance. These test scripts usually test edge cases that are not valuable to customers reading the User Guide.

Test Script Locations

All test scripts are organized in `Library/Documentation/User_Guide`. Within each main folder are Task topics that belong in the User Guide as well as a dedicated test scripts organized into a `Test_Scripts` folder.

Here's an example of our `User_Guide` folder structure:

- User Guide folder
 - Main folder
 - Test Scripts folder
 - dedicated-test-script.dita
 - dedicated-test-script.dita
 - test-script-map.ditamap
 - Subfolder
 - task-topic.dita
 - task-topic.dita
 - submap-example.ditamap
 - Subfolder
 - task-topic.dita
 - task-topic.dita
 - submap-example.ditamap
 - mainmap-example.ditamap

- Main folder
 - Test Scripts folder
 - dedicated-test-script.dita
 - dedicated-test-script.dita
 - test-script-map.ditamap
 - Subfolder
 - task-topic.dita
 - task-topic.dita
 - submap-example.ditamap
 - Subfolder
 - task-topic.dita
 - task-topic.dita
 - submap-example.ditamap
- mainmap-example.ditamap

Creation Process

As new features and bug fixes are developed, Information Developers must create Task topics to explain how to use the new feature or test the bug fix.

Writing Task topics in the User Guide

The task topics in our User Guide are relevant to our users, but also function as test scripts. So we don't burden our users with content that doesn't matter to them. We conditionalize out the quality assurance specific content.

Keep these guidelines in mind when creating Task topics in the User Guide:

- Ensure the Task topic contains a procedure that is relevant to our end users.
- Create the Task topic in the corresponding subfolder it relates to in `Library/Documentation/User_Guide`. For example, if you're writing a Task topic for a review feature, create the task topic in the Review folder.
- Create the Task topic using our Default Task topic template.
- Add the Task topic to the map in the main folder. For example, if you're writing a test script for a review feature, add the test script to the Review map.

Using the Step Result element

Every Step element must have a Step Result element that clearly describes, and if necessary, shows what the result for that step is. The last Step element in a Task topic is the only exception to the Step Result element requirement. Because the topic's Result element is always used in our Task topics, having a Step Result element for the last step would be redundant.

Conditionalizing QA Content

Because every Step element must have a Step Result element, sometimes this means that you end up with a Step Result element that describes things that are too obvious for our end-users. In this case, you can conditionalize that content so that it's excluded in our final output. We use the `userTesting` value in the **Audience** field to conditionalize content meant only for the QA team.

When applying the `userTesting` value, make sure you apply it to the Step Result element and not to the Paragraph element. Conditionalizing just the Paragraph element means, when the `userTesting` value is excluded, only the Paragraph element will be excluded, leaving an empty Step Result element.

Writing Dedicated Test Script Task topics

We have a dedicated `_Test_Scripts` folder in each section of the User Guide to hold test scripts that aren't relevant to our users but which we still need to test. This includes complex or edge procedures that users wouldn't need to perform, for example, a test script to test that an error message displays following specific actions. Because these task topics are meant for testing only, you don't have to conditionalize any of the content.

Keep these guidelines in mind when creating dedicated test script task topics:

- Ensure the Task topic contains a procedure that is not directly relevant to our end users but still requires testing.
- Create it in the dedicated `_Test_Scripts` subfolder in the main folder it relates to. For example, if you're writing a Task topic for checking that REST URLs work, create the Task topic in the `_Test_Scripts` subfolder in the `Resource_Viewer` folder.
- Create the Task topic using our Default Task topic template.
- Add the Task topic to the map in the `_Test_Scripts` subfolder. For example, if you're writing a task topic for checking that REST URLs work, add the task topic in the Test Scripts map in the `_Test_Scripts` subfolder in the `Resource_Viewer` folder.

Using the Step Result element

Every Step element must have a Step Result element that clearly describes, and if necessary, shows what the result for that step is. The last Step element in a Task topic is the only exception to the Step Result element requirement. Because the topic's Result element is always used in our Task topics, having a Step Result element for the last step would be redundant.

Testing Process

All Task topics in the User Guide are test scripts. You're responsible for testing all the Task topics in your assigned folders and their subfolders.

The Task topics in each folder should cover all of our primary test cases. But feel free to test out different combinations of selections and options, such as clicking **Cancel** instead of **OK**, etc.

Keep these guidelines in mind when testing:



- Do not test files that have the status **Obsolete** or **Needs Reevaluation**
- Perform each step listed in the procedure and ensure you get the expected result
- Assign the corresponding Internal QA and Content Defect taxonomy metadata based on your results
- Leave a comment in the topic if it failed, only partially passed, or contains a content defect so that reviewers know why it failed or is defective



Tip: Select a folder and use the **Filters** Tab to see all the Task topics in the folder that need to be tested.

Taxonomy Metadata

After testing all steps in the procedure, you must apply taxonomy metadata tags in two categories based on your results:

| Metadata Category | Taxonomy Metadata Tag | Apply when |
|-------------------|-----------------------|---|
| Internal QA | Pass | Can use the feature without issue.  Important: A test script passes even if there is a content defect. Make sure you tag it as Pass and apply the appropriate Content Defect tag. |
| | Partial Pass | Can use the main feature but there are minor issues with display, placement, etc.  Important: A test script passes even if there is a content defect. Make sure you tag it as Pass and apply the appropriate Content Defect tag. |
| | Fail | An issue occurred during one of the steps and you were unable to use the feature. |
| Content Defect | Information Gap | Topic is missing information. |
| | Outdated | Topic contains outdated information. |