A guide to applying DITA and other authoring best practices to produce topic-based content that end users need to understand to use Amazon tools and services effectively and efficiently

Last Updated: June 2017

# Chapter 1

Information Model Overview

This model is based on the Darwin Information Typing Architecture (DITA) standard, an XML architecture for designing, authoring, publishing, and managing content. This document describes how we’re implementing DITA, and which DITA structures and elements are approved for use in our content. We strongly recommend using only the approved elements listed in the Information Model. Using unapproved elements might yield unexpected or undesirable results.

For more information about DITA, see http://dita.xml.org.

## Authoring in XML

This document contains general information about DITA best practices, as well as detailed information that is intended as a reference for advanced users who might need to troubleshoot an unexpected result or adjust XML code by hand. For day-to-day work, you are encouraged to first consult the CMS wiki for guidance in creating and editing topics.

Our authoring tool, XMetaL, automatically sets up DITA elements and attributes, so you will rarely need to enter them in XML code form. XMetaL prevents selection of certain non-allowed elements and also prevents insertion of an allowed element in an invalid location. In addition, XMetaL provides familiar buttons and wizards for creating tables and lists.

We recommend using a template for each topic. These are pre-populated with basic DITA elements in the order they should appear.

## Why DITA?

DITA provides a standardized structure that is designed specifically for creating technical content. Authoring with DITA:

* Makes content reuse easier, eliminating the need to maintain the same content in multiple locations and reducing rework and the chances of error.
* Separates form from content. You can present the same content in different formats without duplicating authoring effort.
* Adds discipline to the content development process, allowing authors to focus on relevance and accuracy and boosting the quality of the final product.
* Makes editorial deliverables more consistent in structure, terminology, writing style, and voice.
* Reduces costs associated with translation.

## Challenges for new DITA authors

Authoring with DITA requires a different mind-set than traditional authoring.

##### Giving up control over formatting

With DITA, pre-programmed publishing scripts determine the way the content looks when it's published. You will use semantic tags in the file structure to indicate what a piece of content is, and not what it should look like. This allows us to reuse the same content in multiple publications without spending time and effort to tweak the format manually for each publication.

##### Writing modularly

Each DITA topic is narrowly focused on one subject matter or piece of information so that different end products can be assembled modularly. For example, if you are asked to author a Help page, you might create it in the CMS as a series of multiple discrete topics, with each topic covering one of the various information points the page needs to provide. This is a little more authoring work upfront, but down the line it enables easier reuse, easier content discoverability, and lower translation and content maintenance costs, as you might only need to revise one piece of the whole rather than an entire deliverable.

##### Planning for reuse

You probably know that there are pieces of content you will use frequently -- for example, brand names, certain links, or legal disclaimers. Rather than typing them into your topics from scratch every time, take advantage of DITA's library and referencing functions. And get familiar with the existing libraries in the CMS so you know what content is already on hand.

## The DITA standard

This Information Model is based on the DITA standard, which includes:

* A core Document Type Definition (DTD) defining the generic topic information type
* Ten specialized DTDs, only three of which we currently use:
  + Task
  + Concept
  + Reference
* A set of elements:
  + Generic elements that are common to all information types
  + Elements specific to certain information types
  + Elements specific to certain domains or types of subject matter
* DTDs defining the DITA map structures, which allow topics to be assembled and structured for output and other purposes
* Support for reuse strategies to generate multiple outputs from the same files:
  + Conditional attributes
  + Content references (conrefs)
* Transforms (publishing scripts) that can be used in conjunction with Seller Central, Paragon, and Knowledge Center stylesheets and their publishing pipelines to create final deliverables

## Information types

We categorize content into three information types:

* **Tasks:** Procedural or how-to information, such as step-by-step instructions. Answers the question, "How do I...?"
* **Concepts:** Overview and background information that describes a product, interface, or task. Answers the question, "What is...?"
* **References:** Information for users to refer to as needed when completing a task. Answers questions like, "How much...?", "How many...?", “Where is…?”, or "What form...?"

All content must be created in topics that are based on one of these information types. Don’t mix information types within a single topic file. If you need multiple information types to display on the same final product, create multiple topics in the CMS and group them together in a map for publication.

## Basic terms

##### element

Semantic markers that define the start and end of content. All element names begin with a "<" and end with a ">". Every element consists of a start tag, such as <p>, and an end tag, such as </p>. Content is entered between the start and end tags. Content can include words, numbers, and other elements as needed or defined.

##### container element

An element used to group other elements. It does not contain character data or other content. For example, the task information type includes a <steps> container element. You use the <steps> element to group a set of <step> elements.

##### inline element

An element that can exist within another element without introducing a line break. For example, a <uicontrol> can be applied to a single word or phrase within a <p> so that no line breaks occur before or after the <uicontrol>.

##### attribute

A form of metadata that provides more information about an element and its content. In XML code, attributes are defined in an element's start tag. However, you will generally use XMetaL's Attribute Inspector to enter them. Attributes consist of a name and value. Within this document, attribute names are prefaced with an @ symbol (for example, @id).

##### DTD (Document Type Definition)

Documents that provide the rules for an information type in DITA. For example, the task, concept, and reference DTDs all require that you include a <title> element at the beginning of your topic. This rule is defined in the DTD. As an author, you will not be working directly with DTDs.

##### map

A file that is used to define a deliverable. For example, a map of a Help page would list all the related DITA topics that should display on the page when the map is published, in the order they should appear.

##### markup

A broad term used to describe the elements and attributes that are applied to content.

##### nesting

Placing an XML element within another XML element. Elements are said to be nested if their start and end tags occur within the start and end tags of another element. For example, in the code block below, two <li> elements are nested within the <ul> element.

<ul>

<li>First list item</li>

<li>Second list item</li>

</ul>

##### single-sourcing

The practice of storing all the relevant content for one particular DITA topic within the same CMS object for all marketplaces.

##### conditionalization

The practice of using XML-based attributes in a single-source topic to indicate situational differences. We generally use conditionalization to provide different information for different marketplaces within the same topic. When the topic is published, these attributes automatically extract the correct text for the intended destination and viewer.

DITA and our DTDs permit us to set a variety of conditions on content. However, to minimize complexity, we limit conditions to marketplace (country). Exceptions are occasionally made for special circumstances, such as video objects that should appear to viewers only in certain language-switcher situations.

# Chapter 2

Beginning a topic

All DITA topics must include a title and prolog section. These elements are placed above the body of the topic.

## Titles

DITA requires a title for all topics. Topic titles are intended to render as the topic heading in the final output.

The <title> element is the first element within the topic root element (such as <task> or <reference>). Type your text directly into the <title> element, keeping in mind the following coding constraints:

* You cannot nest block elements or most inline elements within <title>. Even when they are allowed, avoid using inline elements that might override the format of the heading style.
* The current structure of Help Builder means that titles will publish as duplicates if you output them as-is to Seller Central. To prevent this, set the @outputclass for the <title> element as notitle.
* If you intend to publish multiple topics on one Help node, you can set the @outputclass for the <title> element as hbtitle for topics with a sortIndex of 2 or more. This will render the topic titles as subheadings on the page.
* Titles cannot be conditionalized in the standard way. If different marketplaces require different titles, use the <ph> inline element within the <title> element and apply a condition to the <ph>. For more information, refer to Conditional Processing.

<title><ph ishcondition="country='United States'">Managing Orders</ph></title>

The content of the <title> element must clearly convey the content found within the topic body. When writing titles, follow these guidelines:

* Make the title specific to distinguish it from other similar topics.
* Use words that accurately describe the content. Be specific, concise, and meaningful. Avoid headings that identify the type of information, but not the subject matter. For example, *Introduction* or *Overview*, without further words does not clarify what content is introduced or what you are providing an overview about. Content titles get the highest priority in the search index making the content easier to find and easier to identify in search results. Content titles also help other authors find relevant information in the repository.
* Use recognizable, search-engine-friendly terms, even if those terms do not appear in the text or product feature.
* Use industry standard and commonly used words.
* Use the same terms consistently in topics that cover the same subject.
* Try to limit the length of the title to no more than eight words, or 55-65 characters (including spaces).
* Use plural forms of nouns unless the subject makes sense only in the singular.
* Avoid using acronyms and abbreviations in a title, unless they are industry standards.
* Use articles (a, an, and the) to improve readability.
* Use sentence case, capitalizing the first word of the title and proper nouns.

## Short descriptions

The <shortdesc> element is not required by DITA but you might run across it occasionally in older topics or templates. We do not recommend using <shortdesc> because this element cannot be easily conditionalized.

If you find content within a <shortdesc> element, move it into a <p> at the beginning of the topic body so that it can be conditionalized as normal. If the topic is a task, you can insert a <p> into a container such as a <prerequisite> or <context> element.

## Prologs and metadata

When you start or modify a topic, add metadata to the <prolog> section to capture information about the topic as a whole. The metadata inside the <prolog> will not be displayed with the topic on output but might be used by processes that publish or translate content, filter content, generate search indexes, or customize navigation. Include each of the following elements in the prolog in the order listed. All elements are required unless noted otherwise in the description.

##### <metadata> metadata

Use the <metadata> element as a container for topic-level metadata. This is prepopulated in our topic templates.

##### <keywords> keywords

Nest the <keywords> element in <metadata> as a container for keywords that can be used by a search engine.

##### <keyword> keyword

Nest the <keyword> element inside <keywords> to contain a term that will be used as a keyword for search engines. You can include as many <keyword> elements as needed.

##### <othermeta> other metadata

Use the <othermeta> element to provide additional metadata information. Use the @name attribute to specify the metadata you are providing and the @content attribute to provide its value. We require several <othermeta> elements:

* **parentId:** The Help Builder node ID of the topic. (Also referred to as a page ID.) Enter the ID in the @content attribute.
* **topicId:** The Help Builder module ID of the topic. (Also referred to as a module ID.) Enter the ID in the @content attribute.
* **language:** This is en\_US on our templates. Do not change, even if your single-source content will not publish in the US marketplace.
* **country:** This is US on out templates. Do not change, even if your single-source content will not publish in the US marketplace.
* **sortIndex:** This number represents the order in which the topic will display on the page. For example, a <sortIndex> of 1 indicates that the topic will be the first module on the page. Defaults to 1 on our templates. Change by adjusting the @content attribute. A value of 0 will not publish. A value of -1 will hide the topic in Help Builder. If there are multiple topics on a single page, make sure the sortIndex for each topic is sequential with no gaps. Duplicate sortIndex will publish with the same number in Help Builder, but the sequence will depend on which topic published first. This could result in topics being out of order.
* **group:** This number corresponds to a module's Groups setting in Help Builder. Groups restrict viewing of Help content to selected sellers, based on account rights and roles. When setting groups, it is easy to accidentally include or exclude sellers outside the target group. Test your settings thoroughly and do not rely on Spoofer to verify the Group experience. You can add multiple <othermeta> elements for groups, and you can conditionalize these elements. If a topic does not include an <othermeta> element for a group, the topic will publish as Visible to All in Help Builder.

### Keywords

All topics should include keywords to help authors and users find information. On processing, <keyword> elements are converted to metatags for HTML output. These metatags assist search engines in assigning a relevance ranking to your topics in a search results list. When the user's search term is found in the metatags, the topic gets a better ranking than if the term is only found in the body text.

There is no limit to the number of keywords you can specify for a topic. However, keep in mind the relevance of the terms as well as translation concerns. That is, keywords in English might not have the same nuance of meaning in other languages when translated.

When selecting keywords:

* Include terms only if there is meaningful content associated with that term. Just because the term occurs in the topic is not a reason to include it as a keyword. For example, many of our topics mention "seller account," but you would not want to list this term as a keyword unless the topic was specifically about using Seller Account tools. Keep in mind the number of hits the user will see for a specific keyword during a general search. Include the keyword only in the most relevant topics to help users narrow their search.
* Include user vocabulary as well as the terms used within the topic, if they are not the same, to increase the likelihood that users will find information. For example, some sellers might use the term "performance metrics" interchangeably with "customer metrics."
* Use the same <keyword> values across topics to ensure consistency in terms that will be referenced frequently. Refer to the TAM SOA/FBA TD (term database) in WorldServer for brand names and other commonly used terms.
* Include synonyms for the important words in the file.

Place keywords in the topic's <prolog> section only. Avoid using <keyword> elements within the body of the topic.

**Note:** Some nodes and language layers in Help Builder feature a standalone, empty module that contains only keywords. It is OK to leave these modules as-is. But as a best practice, keywords should also be incorporated into each topic's metadata for better searchability.

# Chapter 3

Tasks

Task topics provide instructions for completing specific tasks. They typically contain:

* Information immediately required by users to accomplish a single task.
* Minimal information to explain concepts to understand the task.
* Links to related information that can help with executing the task.

They do not provide conceptual descriptions of product features or explanations of how something works, nor do they provide detailed data that might be needed to complete the task. Use concept and reference topics to provide this type of information, even if it means the final output (such as a Help page) will be composed of several topics.

Our task topics generally describe how to use a Seller Central tool to accomplish one specific action. For example, you might create a task topic to outline the steps for:

* Downloading a Payments report
* Resolving an error code
* Using the 1x1 tool to add a product listing
* Using Buyer-Seller Messaging to respond to a buyer's query

## Task structure

Structure your task topics as follows, keeping each element in the order shown. Many of these elements will be prepopulated in our Task template. If you choose to omit an optional element, go to the next element in the structure.

* <task> (required)
* <title> (required)
* <prolog> (required)
* <taskbody> (required)
* <prereq> (optional)
* <context> (optional)
* <steps> (required)
* <step> (required)
* <cmd> (required)
* <result> (optional)
* <example> (optional)
* <postreq> (optional)

## Task elements

Choose from the following elements when writing a task topic. An asterisk (\*) next to the element name indicates that you can nest the block and inline elements that are common to all information types as described in Block Elements and Using Inline Elements.

### Elements within the <taskbody> element

##### <prereq> prerequisite\*

Use the <prereq> element to specify anything that the user needs to know or do before starting the task. For example, you might describe an action that must be completed prior to beginning the task, or list special tools or equipment that the user needs to complete the task. You might also use <prereq> as a container for notes about the entire task.

* Do not use an ordered list to insert a separate set of step actions for prerequisite tasks. Create a new topic for that action and reference it from the <prereq> element.

##### <context> context \*

Use the <context> element to identify the purpose or goal of the task, as well as to indicate to users what they gain by completing the task. The context section should answer questions like "who," "what," "why," "where," and "when," while the steps section then answers the question "how." Although the context information may contain some basic conceptual information, it should not replace or re-create a related concept topic.

##### <steps> steps

Use the <steps> element to create a numbered list of all the steps that the user must follow to complete the task. The <steps> element can contain any number of <step> elements. The <steps> element is required within a task topic.

##### <step> step

Use the <step> element for each step that the user must follow to complete the task. Each <step> element must be nested in a <steps> container element and must include a <cmd> element. You may use one or more <step> elements within each <steps> element.

##### <result> result\*

Use the <result> element to describe the expected outcome for the task as a whole. The <result> element could include a final description using tables, text, or graphics that show end users that they have successfully completed the task.

##### <example> example\*

Use the <example> element in a task to include a specific scenario for completing the task. The <example> element could include graphics or text that show or describe to users how they would complete the task in the given circumstances.

##### <postreq> post requirement \*

Use the <postreq> element to specify anything that the user needs to know or do after completing the task. This information might include recommended next steps, such as cleanup, testing, and adjusting.

### Elements within a <step> element

##### <cmd> command

Use the <cmd> element to describe the action the user needs to take in a <step> or <substep> element. This element is required within <step> and <substep> elements.

##### <info> information\*

Use the <info> element to provide additional information for completing a step beyond the instruction in the <cmd> element or as a container for a <note> element that pertains to the specific step.

##### <substeps> sub-steps and nested <substep> sub-step elements

Use the <substeps> element to break a step down into a series of actions. This element is a container for the individual <substep> elements. Use a <substep> element for each sub-step that a user must follow to complete the step. The <substep> element has the same structure as the <step> element, but it cannot contain another level of <substeps>, <choice>, or <choicetable> elements.

**Note**: If you need to use <substeps>, consider creating a new task instead so that the steps are not embedded in another task. Use of sub-steps should be limited, and only one level of substep is permitted.

##### <choices> choices and nested <choice> choice elements\*

Use the <choices> element to create a list of options that the user may select to complete the task. Use <choicetable> when the user has multiple options that lead to different results or need additional explanation. The <choices> element is a container for individual <choice> elements. Nest a <choice> element within <choices> for each option that the user can select. Do not insert the word "or" between the choices. If necessary, you can add additional elements such as <p>, <ul>, or <ol> to a <choice> element.

##### <choicetable> choice table

Use the <choicetable> element to create a table of options that the user could select to complete the task.

|  |  |
| --- | --- |
| Option | Description |
| <choices> | Create a bulleted list of options. |
| <choicetable> | Create a table of options. |

The specific elements required in a choice table are explained in <choicetable> Elements.

##### <stepxmp> step example\*

Use the <stepxmp> element to include an example illustrating the step. While the <cmd> element is typically written generically to apply to any situation, the <stepxmp> element shows a specific application of the <cmd> statement to help users better understand how to complete the step.

##### <stepresult> step result\*

Use the <stepresult> element to explain the expected outcome of a step. Step results assure users that they are on track, but do not need to be included on all steps. Use this element only as much as necessary to keep the user on task, where it is really useful to the users, and not just to illustrate what they can see on the screen. Remember, if you don't use this element, you can use orienting language in the step itself, such as "On the XYZ screen, do...". Step results are most useful when a <cmd> results in a change in the UI, such as navigating to a new page or a dialog box.

**Note**: You can include only one <stepresult> in a <step>, and nothing can follow it.

### Elements within <choicetable>

XMetaL will set up a <choicetable> element for you without you needing to code most of the individual table elements. However, for reference purposes, the following list describes the elements that make up a choice table. These element labels will not appear in the **Tags On View**. You can either find them in the **Plain Text View** or within the Attribute Inspector.

##### <chhead> choice head

Use the <chhead> element to provide headings for the columns in a choice table. The <chhead> element contains the <choptionhd> and <chdeschd> elements. If you do not specify these elements, the default headings are "Option" and "Description."

##### <choptionhd> choice option head

Use the <choptionhd> element to specify the heading for the first column of a choice table.

##### <chdeschd> choice description head

Use the <chdeschd> element to specify the heading for the second column of a choice table.

##### <chrow> choice row

Use the <chrow> element to create a row in a choice table. The <chrow> element contains the <choption> and <chdesc> elements.

##### <choption> choice option\*

Use the <choption> element to describe an option that the user can choose to accomplish the step. You can use only one <choption> element in each <chrow> element.

##### <chdesc> choice description\*

Use the <chdesc> element to provide a description for the option in the corresponding <choption> element. The description explains why the user would choose that option or the result of making that choice. You can use only one <chdesc> element in each <chrow> element.

## Task authoring guidelines

In addition to structuring your content in a consistent manner using the prescribed elements described in this document, it's important that you also write each section according to specific guidelines so that content can be combined in a variety of ways without it being readily apparent that many different authors contributed to the content.

### General guidelines

Each Task topic should describe a series of steps for one, and only one, action.

* Create a separate Task for each set of action steps you want to describe. Related Tasks can be grouped into the same map to display together when published.
* <substeps> should be true substeps of the primary action and not a separate Task.
* Do not use <ol> elements as a cheat to add more sets of action steps to the same Task topic.

Elements in a Task topic can be conditionalized just like any other element. However, the conditions might not always display in the same way in XMetal as in other topic types.

* Check the @ishcondition field in the Attribute Inspector to verify which conditions are applied to an element in a Task topic. Do not rely on Tag view.
* Set conditions for individual <step> elements. Do not create a new <taskbody> to conditionalize steps for different marketplaces.
* Even if steps are duplicated for conditionalization differences and the step numbers appear to be incorrect in XMetal, the numbering will render correctly for each country when the topic is published.

### Prerequisites

A prerequisite is a requirement that must be met before the user can start the current task. Prerequisites might include:

* Knowledge that the user must possess.
* Other tasks that must be completed before beginning this task.
* Tools required to complete the task.
* Environmental or safety conditions that must be met.

If there are no prerequisites, do not include this element in the topic.

When writing prerequisites, follow these guidelines:

* If the prerequisite is explained in another topic, link to that topic, providing explanatory text that helps the user understand what information they will find if they follow the link and why it's important to the current task.
* You might need to include a "Prerequisites" introduction to the list, using a <p> element.

The <prereq> element is the first element in the <taskbody>, as follows.

<taskbody>

<prereq>

<ul>

<li>...</li>

<li>...</li>

<li>...</li>

</ul>

</prereq>

</taskbody>

### Context sections

Context sections help the user understand the purpose of the task and provide some required background information. If you have nothing more to add, simply leave out the <context> element.

When writing <context> sections, follow these guidelines:

* Use the <context> section to answer questions such as who, what, why, when, and where.
* While the context information may contain some basic conceptual information, it should not replace or recreate a related concept topic. If you need more than one or two paragraphs, consider creating a separate concept topic. You might then reference that concept topic in the prerequisite section of the task topic as information that the user should be familiar with before completing the task.
* Do not include content directly in a <context> element. Instead, surround it with the appropriate block element, such as a <p>. You can include most block and inline elements within the <context> section.
* Try to avoid using the <context> element simply to contain a stem sentence introducing the steps ("To do x, complete these steps:"). This type of introductory stem is frequently unnecessary.

<taskbody>

<context>

<p>...:

</p>

</context>

</taskbody>

### Steps and substeps

The <steps> element is the focus of a good task topic. Although it does not contain any content itself, it is the container element for all individual <step> elements within the task. Similarly, the <substeps> element is a container for individual <substep> elements within a step.

Regardless of whether you are writing a <step> or <substep>, you must include a <cmd> element to describe the action that the user needs to take. Keep in mind the following guidelines when writing this command statement:

* In general, include instructions only for a single user action. However, avoid being too granular. If two simple actions are logically combined, keep them together in one command. Example, "Mark the Yes checkbox, and then click the Save button."
* Write commands in the imperative voice.
* Try to keep the number of steps in a procedure at a minimum. Consider breaking long procedures into smaller task topics, especially if the steps consist of many substeps.
* If you need to orient the user to the location where the action is performed or qualify the step in any way, put that information first before describing the action. Example: "In the upper right corner of the screen, click on the Action dropdown."
* Limit the command statement to one sentence. If you need to provide additional information about the command in order for users to successfully complete the command, use the elements described in Elements within a <step> element.
* Try to include only one level of steps within a task. Although you can enter substeps to provide further detail on how to complete an individual step, too many substeps become difficult for a user to follow. A flat step structure is preferred. Limit your use of substeps to situations when most users will understand how to complete the step from the primary step description alone, but some users will need the additional detail. If most users will need the details, flatten the structure by making each substep a full step in the task.
* Do not use the step/substep structure to provide alternate ways of doing the same task or to provide similar steps for differing conditions. Instead create different topics for each situation.
* Use <choices> when the user has many ways to complete the step to achieve the same result. Use <choicetable> when the user has many options that lead to different results.

### Results

A <result> element enables you to describe to the user what should happen or what something should look like when the entire task has been completed correctly.

Do not include a <result> element if the result is obvious; for example, a simple restatement of the task title.

### Examples

Users can greatly benefit from examples that illustrate the task they are trying to complete or that help them relate a concept to their own situation. Use the <example> section at the end of the task to present specific scenarios and show how the task would be completed in those scenarios. When choosing examples to include, follow these guidelines:

* Include examples for difficult-to-understand tasks and concepts.
* Choose examples that the majority of your users can relate to.
* Choose examples of average complexity, not too simple nor too difficult.
* Choose comprehensive examples that exercise all the steps in a procedure.
* Use SOP materials as a resource for the types of examples that will be useful to sellers.
* Set up the scenario for the example clearly at the beginning of the example.
* Do not include multiple scenarios in one example.

### Postrequisites

A <postreq> specifies anything that the user needs to know or do after completing the task. This information may include actions that need to be completed before the user can see the expected results, such as information that the user needs to read or cross-reference to verify the completion of the task. When writing a postrequisite, follow these guidelines:

* Do not include steps. If the postrequisite is another procedure, link to that procedure instead.
* Do not include obvious actions, such as removing safety equipment or cleaning up the area.
* Include information about the criticality of the postrequisite; for example, does it need to be completed before any other action can be taken?

# Chapter 4

Concepts

Concepts provide background content that helps readers understand essential information about a product, interface, or task. Concepts typically contain:

* Information that answers "What is...?" or "How does this work?"
* Information that is explanatory or descriptions of concepts that are required to complete a task.
* Background or overview information such as purpose, scope, and design of a product, feature, or function.
* Information that is not immediately required by users to accomplish a task.

Concepts do not tell how to use features or functions. Use task topics to communicate that type of information.

Concept topics might include the following categories of information:

* Overviews
* Theory of operations
* Best practices
* Decision support: Helping an end user choose between multiple options
* FAQs. However, we discourage use of FAQs as a way to organize content in long-form deliverables. An FAQ, inherently, is an acknowledgment that we have gaps in the information that is presented. If we are writing topics accurately and comprehensively, we should not have information gaps. See Chapter 14 for guidance on inserting FAQs in long-form Help content.

It can be difficult to decide when to use a concept information type versus a reference information type. For guidance, see Choosing Between a Reference and a Concept Information Type.

## Concept structure

Structure your concept topics as follows, keeping each element in the order shown. If you choose to omit an optional element, go to the next element in the structure. Within the <conbody>, you can use <section> and <example> elements in any order.

* <concept> (required)
* <title> (required)
* <prolog> (required)
* <conbody> (required)
* Any combination of body elements, such as <p>, <ul>, <table>, and so on (optional)
* Any number of <section> elements (optional)
* Any number of <example> elements (optional)

## Concept elements

Nest any combination of the following elements within the <conbody> element.

**Note**: The <conbody> element can also contain any of the block and inline elements that are common to all information types and as long as you have not inserted a <section> or <example> element. As soon as you include one of these two elements, all block elements must be nested within a <section> or <example> element.

##### <section> section

Use the <section> element to organize subtopics in the body of a larger concept topic. A section can have its own <title>, and you can then include any of the common block elements within the <section>. A section <title> will appear as a subheading when published. However, each section can only have one <title>, and the <title> will always be published at the beginning of the <section> content. So, do not place the <title> midway through the <section> and expect the ordering to be preserved.

A <section> must be followed by another <section> or an <example>. Once you insert a <section>, you can no longer include common block elements outside of a <section> or <example> element.

**Note**: DITA does not support nested sections -- that is, you can only have one level of subheading within a topic.

##### <example> example

Use the <example> element to illustrate the content of a topic. The <example> element can include any common block elements. It is recommended, but not required, that you begin each <example> with a <title>. An <example> must be followed by another <example> or <section>. Once you insert an <example>, you can no longer include common block elements outside of a <section> or <example> element. An <example> element cannot be nested inside a <section>.

## Concept authoring guidelines

Concept topics are open-ended, with no required structure. After you have given your topic a title and completed the prolog, you can include any combination of block and inline elements in any order and nested as needed within smaller sections. As you write, keep in mind the following guidelines:

* A concept topic should address one complete idea. In many cases, it makes sense to divide a large subject into many smaller topics, with related links. However, avoid transitional language that assumes readers are coming from one topic to another.
* Use minimalist writing techniques to create content that users can easily and quickly understand.
* Start with general information and gradually add more detailed information.
* Write most conceptual information in paragraphs and unordered lists.

**Note**: When using an ordered list in a concept, ensure you are not writing a task. Ordered lists in concept information typically relate to descriptions of processes in which the user is not the primary actor.

* To allow for as much reuse as possible, avoid transitional words that tie content between paragraphs (for example, "therefore", "furthermore", "as a result", and so on).

### Dividing a concept topic into sections

If the concept you are writing is long and complex, break the topic into sections, each with its own <title>. Keep in mind the following:

* If the individual sections are long or can stand alone without the context of the other information, consider creating multiple concept topics rather than sections within one concept. You can publish all the topics together in the same output.
* Do not include only one section within the topic. If you need only one section, you can likely write the topic so that the material flows from the introductory part of the topic without a section division.

### Concept examples

The <example> element is a special type of section used to provide a context that might help users understand the concept. When creating examples, keep in mind the following:

* The <title> element within an <example> is optional.
* Include only one example per <example> element.
* Try to include examples that will apply to a wide range of users.

# Chapter 5

References

Reference topics provide look-up, data-oriented information. References are used to define specific, often detailed information, or to bring together "at-a-glance" information (tables) that users refer to as required. References do not typically include descriptions or explanations.

Reference topics include the following types of information:

* Fee lists
* Error codes
* Template columns and attributes
* Tabular data charts, such as links to flat files
* Specifications

It can be difficult to decide when to use a reference information type versus a concept information type. For guidance, see Choosing Between a Reference and a Concept Information Type.

## Reference structure

Structure your reference topics as follows, keeping each element in the order shown. Within the <refbody>, you can use <table>, <section>, and <example> in any order and intermixed as needed.

* <reference> (required)
* <title> (required)
* <prolog> (required)
* <refbody> (required)
* One or more <table> elements (optional)
* One or more <section> elements (optional)
* One or more <example> elements (optional)

## Reference elements

Nest any combination of the following elements within the <refbody> element.

##### <table> table

Use a <table> element to display tabular data. For more information on the elements that comprise a fully defined <table>, see Building Tables.

##### <section> section

Use the <section> element to organize subtopics in the body of a larger reference topic. Begin a section with a <title>. You can nest any of the common block and inline elements within the <section> as described in Block Elements and Using Inline Elements. You cannot include these elements (except for tables) outside of a <section> or <example> element in a reference topic.

##### <example> example

Use the <example> element to illustrate the content of a topic. It is recommended, but not required, that you begin each <example> with a <title> unless it is the first element in the topic. The <example> element can include any of the common block and phrase elements within the <example> as described in Block Elements and Using Inline Elements. You cannot include these elements (except for tables) outside of a <section> or <example> element in a reference topic.

## Reference authoring guidelines

Reference topics are meant to be consulted quickly so that the user can get the information that they need about a particular setting, specification, or feature and get back to performing a task. Because the user scans a document for this small bit of information quickly, best practices dictate a minimalist construction consisting of consistent, well-segmented data that is labeled in user terms. When writing reference topics:

* Avoid long blocks of narrative information in <p> elements. You can assume that users of reference topics already understand the basics. You could also consider creating a separate concept topic for lengthy background information.
* Make information easy to scan by presenting information in lists, tables, or charts.
* Because this information is not expected to be read from beginning to end, there is no need for transition sentences or other structures for making information flow from one section to the next.

### Choosing between a reference and a concept information type

|  |  |
| --- | --- |
| Use a concept information type when | Use a reference information type when |
| You are providing explanations. | You are providing data. |
| You expect the user to read this information only once or twice before internalizing the content. | You expect the user to refer to this information on a regular basis. |
| You expect the user to know this information before completing a task. | You expect the user to need this information during a task. |

# Chapter 6

Block elements

Block elements are common to all DITA information types. Nest these elements within the topic body element (<taskbody>, <conbody>, and <refbody>) to create the basic topic structure. Block elements include:

* Paragraphs
* Lists
* Tables
* Images
* Notes

## Paragraphs

The paragraph (<p>) element is the most common element used in topics. Write concise paragraphs so that users can easily scan and process the topic. Avoid long walls of text.

* Limit each paragraph to one idea.
* Keep paragraphs short and to the point. A good goal is three to five sentences.
* Break up long blocks of text with lists, tables, and other visual elements between paragraphs wherever possible.
* Start paragraphs with more general information and move to the specific.

### Introductory paragraphs

Many paragraphs provide introductory information to other DITA elements, such as lists, notes, and tables. When writing these paragraphs, they should stand alone from the following element. Do not nest the list, note, or table within the introductory <p>. Nesting these elements can cause weblab errors in the HTML output for Seller Central.

### Including <p> elements in other DITA elements

Include <p> tags on all paragraphs, even if an individual inline element has only one line of text. This approach allows us to add conditions to the content.

## Lists

Lists can help users scan for and find information quickly. When written correctly, they present information in a logical order, in small, easily understood pieces.

You do not need to use any list element if you have a list of three or less, single-word items.

<p>We uses three list types: unordered, ordered, and definition.</p>

When writing lists, keep in mind the following guidelines:

* Write lists in a parallel structure. For example, if one list item starts with a verb, make all the list items do so. If one list item is a complete sentence, make all the items complete sentences.
* Include all common information from the list items in the introductory sentence to avoid repetition.
* If the list has an introductory statement, close the <p> element before beginning the list.
* Use complete sentences to introduce the list. Do not make the sentence grammatically dependent on the list items by completing the sentence with them.
* Do not create single-item lists for the sake of extra indenting or formatting on publication.
* Unless the list appears in a reference topic, try to limit the number of items to fewer than nine to help prevent walls of text on output.

### Unordered lists

Use when the sequence of the list items doesn’t matter. An unordered list begins with a <ul> element and contains any number of nested <li> elements formatted with bullets.

**<ul> unordered list**

Start an unordered list with the <ul> element. This element is the container for all unordered list items; it does not contain any text itself.

**<li> list item**

Use an <li> element for each item in the list. There is no limit to the number of list items you can include. You can nest other block elements, such as paragraphs and notes within the <li> element, but their use should be limited. Do not place a <p> at the beginning of an <li>. Only use the <p> in an <li> when you need to begin a new line of text.

When writing unordered lists, follow these guidelines:

* In general, list items should be short so the list can be scanned. If you are including more than two sentences in your list items, consider whether the information would be better included in another way.
* Arrange the list items in a logical way, such as alphabetically. If necessary, provide an introductory explanation about how the list is organized.
* Indicate how the list items relate to each other; for example, are they mutually exclusive items, where only one can be true at a time, or can more than one or all apply to the situation.
* Ensure that it is clear whether the list is comprehensive, showing all available options, or only a subset of the possibilities. For example, the words "including" or "such as" imply the list is not complete, while the words "if applicable" imply the list might contain more items than apply to the user's specific situation.

### Ordered lists

Use when the sequence of list items is important. An ordered list begins with an <ol> element and contains any number of nested <li> elements formatted with consecutive numbers.

**<ol> ordered list**

Start an ordered list with the <ol> element. This element is the container for all ordered list items. It does not contain any text itself.

**<li> list item**

Use an <li> element for each item in the list; there is no limit to the number of list items you can include. You can nest other block elements, such as paragraphs and notes within the <li> element, but their use should be limited. Do not place a <p> at the beginning of an <li>. Only use the <p> in an <li> when you need to begin a new line of text.

When writing ordered lists, follow these guidelines:

* Do not use an ordered list to document task steps within a concept or reference topic. Instead, use a task topic and <step> elements.
* Avoid using imperative verbs in ordered lists. The use of an imperative verb implies you are documenting a task, and you should be using a task topic instead of an ordered list.
* Do not use an ordered list when the stem sentence refers to the specific number of items in the list. Only use the ordered list if the sequence or priority of list items is important.

### Definition lists

Use when defining or comparing a series of items. Some topics use <dl> for FAQ lists because the structure is easier to conditionalize.

A definition list begins with a <dl> element and contains any number of nested <dlentry> pairs, consisting of a <dt> element and a <dd> element:

**<dl> definition list**

Start a definition list with a <dl> element. This element is the container for all elements in a definition list. It does not contain any text itself.

**<dlentry> definition list entry**

Use the <dlentry> element in the <dl> element as a container for each term and definition pair. Insert a separate <dlentry> element each time you introduce a new pair.

**<dt> definition term**

Nest a <dt> element within a <dlentry> to identify the word or phrase to be defined. If multiple terms share the same definition, you can add multiple <dt> elements within a single <dlentry> element. The <dt> element must precede the <dd> element within each <dlentry>.

**<dd> definition description**

Nest a <dd> element within a <dlentry> to provide the definition for the corresponding term. If a term has multiple definitions, you can add multiple <dd> elements within a single <dlentry> element. You can include any common block elements, such as paragraphs or other list types, within the <dd> element. You must include at least one <dd> element—if a <dd> element is missing, you will receive an error when publishing.

When writing definition lists, following these guidelines:

* Do not use a <dl> as a way of nesting "sections" in <section> elements. If you require more than one subtitle level, this is an indication that you should split the topic into multiple subtopics.

### Nesting lists

To create sub-lists, nest list elements in other list elements. You can nest any type of list in any other type of list. Be aware, however, that nesting more than one level of list elements can make the list more difficult for end users to understand.

When lists are conditionalized, nesting is also very difficult for future authors to decipher and is not a conditionalization best practice. Use nesting sparingly.

### Reusing list items

To allow for the possibility of reusing list items in other locations, follow these guidelines:

* Do not insert a list in the middle of a sentence.
* Do not write list items to complete an introductory sentence.
* Include closing punctuation on list items only if all items in the list are complete sentences. This maximizes reuse potential when the context of a list item is not known.
* You can nest block elements, such as paragraphs and notes in list items, but limit their use to keep the list as simple, reusable, and scannable as possible.
* Capitalize the first word in each list item, unless the word should never be capitalized such as a list of commands.

## Notes

Use the <note> element to call attention to important information, best practices, and tips that might be overlooked in the text. Notes might emphasize a point, or contain an explanation or comment.

### Best practices for writing notes

With notes, less is more. Users become conditioned to ignore admonishments when there are too many competing for attention. Ensure that the text truly needs to be emphasized.

* Ideally, notes should be no more than one or two sentences. For anything longer, consider whether some of the information could be incorporated elsewhere into the topic content instead.
* Similarly, try not to insert <ul>, <ol>, or other block elements into a note unless there is a compelling reason to do so.
* Limit the number of notes you place within one topic. Keep in mind, too, that a single publication might be composed of multiple topics, all of which might or might not contain notes.
* For consistency, choose only the note lead-ins on the "recommended" list. These are Note:, Important:, Tip:, and Warning:.
* Do not place notes above all the rest of the content in a topic unless the information in the note is of high importance. Situations where notes at the top of a topic are appropriate would include a mandatory legal disclaimer on a terms page, or an announcement of an upcoming policy change that will affect large numbers of readers.

### Note element and attributes

Use the @type attribute to indicate the type of note you are including. This attribute defines the lead-in that will appear with the note. Do not type the lead-in text separately into the content, or repeat the defined lead-in.

When adding a note and the content is relatively general (for example, the note that a particular Seller Central feature is available only for Professional sellers), always check the repository Library folder first for an existing version and use a conref to reference it into your topic (see Reuse Strategy). If an appropriate statement does not exist, write the text and submit it for inclusion in the repository for future use.

### Note type definitions

Recommended:

* **Note:** Default choice, will display if you leave the @type attribute blank
* **Important:** Indicates a piece of information the user needs to know before proceeding.
* **Tip:** Indicates a brief, helpful piece of information that will assist the user in completing a task or understanding a concept.
* **Warning:** Indicates a situation which, if not avoided, could result in negative consequences for the user.

Not recommended, but included here for reference as part of the DITA standard:

* **Attention:** Used for emphasis.
* **Caution:** Indicates a situation which, if not avoided, could result in negative consequences for the user.
* **Danger:** Indicates a situation which, if not avoided, could result in serious harm for the user.
* **Fastpath:** Do not use, even though it is an option in the attribute dropdown.
* **Notice:** Used for emphasis.
* **Other:** Allows you to insert a customized lead-in. Enter your lead-in text in the @othertype attribute.
* **Remember:** Indicates a brief piece of information that should not be overlooked when completing a task or reviewing a concept.
* **Restriction:** Indicates a limitation of some sort. Use sparingly, as this word has a negative connotation.

## Images and media content

Images and media objects can supplement text in your topics. These elements might include:

* screen captures
* icons
* graphs
* illustrations
* schematics
* videos
* animations

Because of the complexities of localizing graphics and media content across all the language-marketplace combinations that our deliverables support, we do not recommend their use. Challenges include:

* **Complexity of file management:** To fully localize an image or video, dozens of copies must be separately created, stored, moved to a publishable workflow status, inserted into topics, and correctly coded for proper display.
* **Complexity of translation management:** Images that are not properly file-managed will block topics from going through the translation pipeline, resulting in delays.
* **Complexity of change management:** UIs change frequently, quickly making older screenshots and video tutorials obsolete and potentially misleading to end users. Because individual product teams also own the media related to their content, there is no standardized mechanism in place to ensure that images and videos stay fresh.

### Guidelines for using images and media

Only include images and other media elements in a topic if they provide a value to the content that a text-based description cannot.

* Aim for media content that does not need localization. That is, there is no visible wording or lettering, it doesn't picture anything that could be culturally sensitive in any marketplace, and it does not illustrate something that could be different for different Amazon sites or marketplaces.
* Product teams own the responsibility for creating all media files, including localized files. They also own the responsibility of providing revised files when UIs change or the content needs updating.
* Callouts or similar markup should not be placed on top of images. These pose translation and localization problems.
* Authors should reject media files that are blurry, overly dark, illegible, or otherwise poor-quality.
* All images embedded in content should have alt-text specified for accessibility purposes.

### Static graphics

Nest static graphics, such as screen captures and icons, in <image> elements. There are two methods for nesting, depending on where the image file is stored.

* **Image stored inside the CMS:** Use XMetaL's tools to locate and point to the correct image file.
* **Image stored outside the CMS:** Use the @href attribute to point to the URL of the appropriate file. This method is preferred because file management is easier. When using this method, always set the @scope attribute to "external."

Use XMetaL's Image Properties tool to provide alt-text that displays for the end user if the image is not displayed. If coding in XML, you can embed an <alt> tag within the <image> element to specify alternate content.

Images can be set inline with text, or to break to the next line. Use XMetaL's Image Properties tool to indicate this, or set the @placement attribute to "inline" or "break."

In general, do not use attributes to set image size, as you cannot guarantee quality across a broad range of screen sizes. The image will default to actual size if you leave the @height and @width blank. If you do set a size, only do so if an image is too large and you want to make it smaller. In this situation, use only numbers for the attribute value, and not units of measurement. For example, width=480, instead of width=480 px.

If your image needs a caption, use the <fig> element instead of the <image> element.

### Videos and animations

To embed media files, embed an <object> element within a <p> or <section> element and specify the following attributes:

* **@classid:** Use "sellertutorial" for Help content.
* **@data:** Provides the location or URL of the video file.

In some circumstances, you might want to specify the size of the video player. Do this by setting the @height and @width attributes. Use numbers only, and not units of measurement. For example width=480, instead of width=480px.

An alternative to embedding a video player is to instead insert an <xref> with a link to the URL where the video is hosted externally, such as YouTube. This approach works best for videos that aren't owned by Amazon.

When one country offers a video in several languages, you have two options to make sure the right language displays based on the user's language-switcher settings:

* Use a series of <xref> elements and link to the individual URLs where each language version is stored.
* Use compound conditions to set both a country condition and a language-code condition on the element housing the video object.

<p country='United States' and language\_code=en-US><object>Video in English</object></p>

<p country='United States' and language\_code=zh-CN><object>Video in Chinese</object></p>

### Graphic placement and formatting

You have very limited control over the placement or formatting of an image or media element using your DITA tags and attributes. Seller Support stylesheets will place and format graphics according to the following rules:

* By default, a figure will be placed within the current text margins. For example, if it is contained within a paragraph, its left side will align with the left edge of the text. If it is in a list, it will be indented to the same left-hand margin as the list element.
* If the image is smaller than the column width, it will be still be left-aligned with the text.
* XMetaL's Image Properties appear to allow you to set horizontal placement, such as centering the image on the page. However, the output on Seller Central is not guaranteed, as the transform sometimes does not actually center the image.

**Note**: Use graphics tools to size images appropriately when they are created. Do not attempt to scale your image using DITA. This can result in poor quality. In addition, it can be difficult to choose the "correct" size because you cannot control the screen sizes where an end user is seeing the image.

## Tables

Use XMetaL's table features to create tables within your topics. It will automatically include all the DITA elements required for each row and column.

### Table text

Tables enable you to organize information into an easily scannable format. Follow these guidelines to ensure maximum usability:

* Write content in cells of the same column in a parallel structure.
* Arrange the table in a logical fashion according to the content of the first column. If the order is not immediately apparent, preface the table with instructions for its use and interpretation.
* Ensure column headings clearly represent the type of content they contain.
* Keep cell contents brief and to the point. In general, avoid writing full sentences in table cells, or including long paragraphs within the table.
* If all information in a column is the same for every row in the table, consider presenting the information in a different way.
* You can include most block and inline elements in a table cell. However, do not nest tables within tables.
* For blank cells, insert a <p> element containing no content. This will prevent unicode characters from outputting erroneously in Seller Central when the table is rendered.

### Add table notes to tables

If a table requires notes to clarify its content, consider using the <fn> element to add these notes. Embed the text that you want to appear in the footnote and the tag at the point where the footnote reference will occur.

**Note**: Table notes that are created by using <fn> will be published at the bottom of the topic. If there is other content between the table and the footnotes and if it is important for the table notes to appear adjacent to the table, do not use <fn>.

### Format table cells

Seller Support stylesheets control most table formatting. However, you might need to use XMetaL's table tools to control alignment of text within the table. These alignments can also be set through attributes.

* Use the @align attribute on the <entry> element to control the horizontal alignment of the text in the column. You can choose between left, center, right, and justify. Left is the default.
* Use the @valign attribute on the <entry> element to control the vertical alignment of the text in the row. You can choose between top, center, or bottom.
* Non-default cell alignments are currently not guaranteed to output as intended in Seller Central. Run a preview of the content in Ajuda Publish to verify.

Column widths and row heights will adjust automatically depending on the content within their cells. Do not adjust table widths manually in an effort to make the table "prettier" or to remove what appear to be large gaps of space. Remember that what looks right (or wrong) on your screen might not look the same across a broad range of end-user screens and sizes.

### Spanning rows or columns

To span a cell across multiple rows, use XMetaL's table tools to Merge cells.

### Table elements

XMetaL will automatically include the appropriate table elements when you insert a table. This section is provided for reference and troubleshooting purposes.

##### <title> title

Adding a <title> to your table is optional.

##### <tgroup> table group

Use the <tgroup> element to specify the display properties for the columns, rows, spanning, header, and body of the table. You must define the number of columns in the table using the @cols attribute. The <tgroup> element is the container for the <colspec>, <thead>, and <tbody> elements.

##### <colspec> column specification

Use the <colspec> element to specify the column information, such as column name, number, cell content alignment, and column width. The column name, number, cell content alignment, and column width are all specified using attributes in the <colspec> element:

* Use the @colname attribute to specify the column name.
* Use the @cols attribute to specify the number of columns.
* Use the @colwidth attribute to specify the width of each column. Column widths should be expressed as relative values, meaning that they are sized relative to each other rather than given a fixed width. There may be several unique cases in the documentation where fixed column widths are used, but this is rare.

**Note**: Determining the acceptable column widths frequently takes some experimentation with these colwidth values, especially if you have specific content that you do not want to wrap within the column. It is not recommended to set @colwidth manually.

##### <thead> table head

Use the <thead> element to define the heads for the table columns. This element is optional, but must appear before <tbody> if used. The <thead> element can contain one or more <row> elements.

##### <tbody> table body

Use the <tbody> element to define the body of the table. The <tbody> element can contain one or more <row> elements.

##### <row> row

Use the <row> element to define a single row in the table. Each <row> element contains multiple <entry> elements, each indicating a different column. The number of <entry> elements in the body of the table should match the number of <entry> elements in the <thead> element, unless you are spanning columns.

##### <entry> entry

Use the <entry> element to define a single cell in the table. The number of <entry> elements defined in the <tbody> element must match the number of <entry> elements defined in the <thead> element. You can enter content into the <entry> element directly. If necessary, you can add additional elements such as <p>, <ul>, or <ol>.

**Sample 4-column table, XML**

<table>

<tgroup cols="4">

<colspec colname="c1" colnum="1" colwidth="1\*"/>

<colspec colname="c2" colnum="2" colwidth="1\*"/>

<colspec colname="c3" colnum="3" colwidth="1\*"/>

<colspec colname="c4" colnum="4" colwidth="1.5\*"/>

<thead>

<row>

<entry>Column 1</entry>

<entry>Column 2</entry>

<entry>Column 3</entry>

<entry>Column 4</entry>

</row>

</thead>

<tbody>

<row>

<entry>Data</entry>

<entry>Data</entry>

<entry>Data</entry>

<entry>Data</entry>

</row>

</tbody>

</tgroup>

</table>

# Chapter 7

Using Inline Elements

Inline elements, also called phrase elements, describe the words or phrases that occur inside a block element such as a paragraph, list item, or table cell. The information contained in these elements is used to optimize search and retrieval for both authors and users. In addition, some inline elements have specific formatting requirements that are handled by the stylesheet.

**Note:** Do not select an inline element based on how it will be rendered, but on what type of information it contains.

## Considerations for translation

The way inline elements are applied to content can affect how the phrases are segmented in translation tools. When segmenting goes awry, it can cause headaches for translators. It can also inadvertently invalidate the XML and cause translations to get stuck in our automated pipeline.

* Sentence punctuation such as commas and periods should be placed outside the tags for inline elements.

<p>Press <uicontrol>Save My Changes</uicontrol>.</p>

* You can also use inline elements to mark content that you do not want translated. Set the attribute @translate to "no" for any element that you do not want translated.

## Programming and software elements

Use these inline elements when documenting APIs, programming, or scripting languages:

| Inline element | Description | Output |
| --- | --- | --- |
| code block <codeblock> | Use <codeblock> to set apart code or system messages from the main flow of the text. Within <codeblock>, line endings and spaces are preserved as entered. | Publishes in Seller Central as monospaced font, no shading. |
| code snippets <ph> or <p> | Use <ph> for code examples that appear within the main flow of text. Use <p> for longer code blocks that don't require specific line breaks. Set the @outputclass as "code." | Publishes in Seller Central in monospaced font with yellow shading behind. |
| variable name <varname> | Rarely used. Use <varname> to define a placeholder for information supplied by the user. | Publishes in Seller Central in italics. |
| user input <userinput> | Rarely used. Use <userinput> when referencing a key name or text the user needs to type. | Publishes in Seller Central as monospaced font. |
| system output <systemoutput> | Rarely used. Use <systemoutput> to display command line prompts, error messages, or other output shown on the screen from the software. | Publishes in Seller Central as monospaced font. |

## User interface elements

Use these inline elements when referring to elements within the graphic user interface:

| Inline element | Description | Output |
| --- | --- | --- |
| user interface control <uicontrol> | Use <uicontrol> to mark up any item that might appear on a screen, such as a window title, button label, menu option, or field name. | Publishes in Seller Central as bold text. |
| file path <filepath> | Rarely used. Use <filepath> to mark up both file paths and file names. | Publishes in Seller Central as monospaced font. |
| menu cascade <menucascade> | Rarely used. Use <menucascade> to present a series of menu choices. The <menucascade> element contains two or more <uicontrol> elements. On output, the content of the <uicontrol> elements will be separated by an angle bracket (>). Be cautious about using this tag as translation tools sometime do not read the auto-generated > correctly and can garble the text. | Publishes in Seller Central as bold text. |
| window title <wintitle> | Use <wintitle> to mark up the title of a window, dialog box, tab, or wizard page name. | Publishes in Seller Central as bold text. |

The table below lists the appropriate inline element to use when documenting common Seller Central UI elements.

|  |  |
| --- | --- |
| Interface object | DITA inline element to use |
| Non-hyperlinked page titles, such as Manage Inventory | <wintitle> |
| Icon names | <uicontrol>  To include the actual icon, use <image> |
| Screen buttons | <uicontrol> |
| Field names, radio or checkbox names, dropdown menu selections, column headers, actions, labels | <uicontrol> |

## Emphasis elements

We allow use of the following non-semantic formatting tags in some situations:

* <b> bold
* <i> italics
* <u> underline
* <sub> subscript
* <sup> superscript

Use <b>, <i>, or <u> sparingly, and only when absolutely necessary for emphasis. These tags are not to be used as a workaround for general formatting. Whenever possible, use a semantic tag instead. When you see bold and italics used in draft content, look for ways to structure this in a more semantic way. For example, if a term is bolded to call attention to its use as a UI label, tag it with <uicontrol> instead of <b>.

If your content will be translated, avoid using <sub> or <sup>. These elements cause problems with segmenting the content for translation tools and can lead to garbled text.

## Miscellaneous inline elements

Use these inline elements to specify special processing of the content they contain:

| Inline element | Description | Example |
| --- | --- | --- |
| <cite> | Rarely used. When referencing another document, enclose the name of the document in <cite> tags. | <p>Refer to the <cite>Chicago Manual of Style</cite> for comma guidelines.</p> |
| <tm> | Use the <tm> element to mark up a trademarked phrase or name.  Use the @tmtype attribute to specify the type of trademark:   * registered marks (reg) * service marks (service) * trademarks (tm)   The stylesheet will automatically include the appropriate mark. |  |
| footnote <fn> | Use the <fn> element to create footnotes in your topics. The stylesheet will automatically add a superscripted number to the referenced piece of content, and the text of the <fn> element will float to the bottom of the page. |  |
| <draft-comment> | Use <draft-comment> to provide feedback, suggestions, or clarification to the reviewer or author of a topic.  The <draft-comment> element is not translated or included in the final output, but can be seen on draft versions of the content.  When entering a draft comment, identify yourself in the @author attribute and a date in the @time attribute. You should resolve and remove all comments before final publishing.  Avoid entering lengthy draft comments or multiple draft comments in a single file. | <draft-comment author="dms" time="29-11-2012"> Which port should the user choose?</draft-comment> |
| <required-cleanup> | Use <required-cleanup> to provide feedback to another reviewer or author that must be addressed before the topic moves to the next step in the workflow. | <required-cleanup>The EU currency conditions don't make sense - UK needs the figure in pounds, and the other locales need euros.</required-cleanup> |
| XML comment <!-- comment here --> | Use the XML code comment to enter comments where a <draft-comment> is not allowed by DITA or where you want to insert a permanent message for future authors. You can also use this element to hide content within your file that is not ready for publication. Content within the XML comment is not processed by the rendering engine when you publish a document.  In XMetaL, access this by highlighting Insert > Advanced > Comment. Note: This will convert elements and attributes to XML code in Tags On View mode. To restore the tags, strip out the XML code comment tags in Plain Text View. | <!-- Begin automated workflow --> |
| <ph> | Use the <ph> element to organize content for reuse or conditional processing.  Use when a <title> needs to be conditionalized. They are also used to create inline buttons and other rare uses. Do not use to conditionalize words or phrases within a <p> or other element. | <section><title><ph ishcondition="country=Spain">Add a Product</ph></title>  ...  </section> |

## Special characters

You cannot enter an ampersand (&) or less-than sign (<) directly in your XML code or in attributes that you set, such as href URLs. These characters indicate to XML that they will be followed by XML entity definitions or elements and can cause validation errors when you try to save your topic. To enter these characters, either ensure you are in the author view and type them directly (for regular text content), or use the following XML coding (for attributes ):

* & &amp;
* < &lt;

You might also find the following codes useful:

* ' &apos;
* non breaking space &nbsp;
* " &quot;
* > &gt;

Other special characters do not have these "friendly" HTML codes, but require you to enter a special numerical code.

XMetaL has many special characters available through a character map. However, to enter these characters manually or to enter characters that are not provided by XMetaL, enter Plain Text View and do one of the following:

* Enter the ASCII code for that character in the form &#nnnn;, where nnnn is a decimal code for the character. For example, &#949; produces the lowercase Greek letter omega ( ω ).
* Enter the hexadecimal code for that character in the form &#xABCD;, where ABCD is the hexadecimal code for the character. For example, &#x221a; produces a square root sign ( √ ).

You can find ASCII, hexadecimal, and HTML code tables on the Internet; for example, <http://www.ascii-code.com/>.

## Elements that we don't use

These elements are part of the DITA standard but are not used by SECM, primarily because they cause conflicts with translation tools:

* **<q>** - Might be seen to indicate material placed in quotation marks. If you see it, remove the tag and type in the punctuation.
* **<menucascade>** - The ">" that is inserted automatically with this tag does not render correctly in some languages. Instead, use a series of <uicontrol> elements and type in the ">" manually between them.

# Chapter 8

Maps

Usually built as the first stage of a deliverable, maps provide a framework for organization of topics and other content outputs. Maps can specify topic sequence and hierarchy and can connect to existing topics (for reuse), other maps, and non-DITA resources such as PDF files. Maps have a .ditamap file extension.

We currently use maps for translation batching, but not for content outputs. Some authors also use maps to keep track of frequently accessed topics without having to search the repository.

## What are maps?

Maps can define an entire deliverable and its subparts. Use maps to:

* Organize related topics into single reusable units
* Organize topics that all relate to a single user goal
* Divide large sets of related information into smaller, more manageable units

Temporary maps can also be built as a work aid for reviewers, translation administrators, and other authors even if these topics will not be referenced together in the final deliverable.

## Map structure using XML elements

Most map structures depend on the specific content in the particular deliverable. You create that structure by combining and nesting map elements as needed. You must begin your map with a title, but all other elements can be interspersed as needed.

* <map> (mandatory)
* <title> (mandatory)
* <topicmeta> (optional)
* Any number, at any level, and in any order of the following:
  + <topichead>
  + <mapref>
  + <topicref>

### Map elements

Nest the following items in the <map> base element when creating a map.

**<title> title**

Use the <title> element to define a title for the group of topics; for example, a Help page heading.

**<topicmeta> topic metadata**

Use the <topicmeta> element to specify metadata about the map as a whole, a group of topics referenced within the map, or a specific topic or map reference. Metadata specified in a <topicmeta> element cascades to all elements contained within the element to which the <topicmeta> is attached. For example, metadata within a <topicmeta> element that appears immediately after the base <map> element applies to all elements in the entire map. However, <topicmeta> attached to a <topicgroup> applies only to the topics within that group, and <topicmeta> attached to a single <topicref> applies only to that topic. Metadata specified in <topicmeta> within a map overrides any metadata values for the same elements specified in the topics, unless multiple elements of that type are allowed. See Map Metadata for the metadata you can define within a map.

**<topicref> topic reference**

Use the <topicref> element to point to a single DITA topic. You must include the @href attribute to point to the path and file name for the DITA topic or external file you want to include. Use multiple <topicref > elements to point to multiple topics to build your deliverable. Nesting the <topicref> elements creates a hierarchical structure.

**<mapref> map reference (optional)**

Use the <mapref> element to point to a map of DITA topics. You must include the @href attribute to point to the path and file name for the DITA map you want to include.

**<topichead> topic head (optional)**

Use the <topichead> element to provide a title-only entry for a table of contents and within the rendered content.

To define the title, nest a <navtitle> element within a <topicmeta> element within this element.

<topichead>

<topicmeta>

<navtitle>Replacing BTS3900 Function Components</navtitle>

</topicmeta>

</topichead>

### Map attributes

Map attributes specify whether topics referenced within the map should be included in various outputs or generated content.

**Note:** The attributes in this table will generally not be used in Seller Central content. They are listed here for reference should a fringe case arise.

| Name | Description | Available Values | Required |
| --- | --- | --- | --- |
| @print | Specifies whether the topic should be included in print outputs, such as PDF. The default is yes. | yes | no | Only if you want to exclude the topic from the print output. |
| @toc | Specifies whether topics are included in a site map or table of contents. The default is yes. | yes | no | Only if you want to exclude the topic from the table of contents. |
| @chunk | Specifies that all nested topics be combined together as a single topic when processed. | to-content | Only if you want multiple topics to be displayed together as one topic in an outline output. |

## Creating maps

A map file consists of a hierarchical organization of <topicref> and <mapref> elements. Each of these elements requires an @href attribute that points to an individual topic or map.

When building a map, consider the following strategies:

* Start by adding task topics, which should be the primary focus of your documentation according to minimalism principles. Consider how best to organize these tasks based on your audience. For example, you might order the tasks based on workflow or the sequence in which the tasks might be completed; based on frequency of use from most common to least common; or based on who will be performing the tasks. Then, add concept and reference topics that support the tasks. Do not add concept or reference topics that you cannot relate to at least one task.
* You can include a topic more than once within a map. However, based on our current outputs, this will likely result in duplicate content for the end user.
* Just because you have created multiple topic files doesn't mean that they need to be displayed as multiple topics. Sometimes you may have created multiple topics for reuse purposes, but the end user should see all the content as one comprehensive topic. Specify @chunk="to-content" on the parent topic to combine it with all nested children into one topic on output.
* Rather than nest topics several levels deep, consider creating submaps that you then reference with a <mapref>. In general, it's a good idea to create submaps whenever you are nesting more than five topics to keep your maps clean and easily interpretable by others.
* Avoid nesting topics more than four levels deep. Although you can nest maps in maps, keep in mind the levels you are creating by nesting maps in maps.
* A table of contents is automatically built when you process a map. To keep a topic from appearing in the TOC, set the @toc attribute to "no".

## Map titles

In general, follow the same guidelines for writing map titles as for topics (see Titling topics). In addition, however, consider the following:

* Map titles will tend to be less specific than topic titles because they are summarizing the content of all the topics nested within the map. For example, a map title for a Help page would be the title of the page, and not the title of one of the individual topics that comprise the page. Think about how you might categorize the information in the map. Often that category should be part of the map title.
* Ensure the title indicates the range of topics it contains. For example, if the content covers both removal and replacement procedures for a board, it would be incomplete and misleading to title the map "Replacing boards."
* In related segments of content, map titles might be closely tied together. Each will deal with some aspect of the overall topic. Therefore, be careful to make each map title clearly differentiated in terms of the specific contents it contains compared to others in the same output.
* Avoid starting all map titles with the same words, such as a feature name. Users are scanning for words that help them differentiate between the maps. Put those words first.

# Chapter 9

Linking Strategies

Although topics in a topic-based structured authoring environment are meant to stand alone, they may still be enhanced by other related content in another topic. DITA provides sophisticated tools for defining and displaying links to related content. But our publication outputs are more limited in scope, so our linking strategies are generally confined to:

* Using inline cross references within a topic
* Displaying dynamically generated links to subtopic nodes on Help pages in Seller Central. No author or DITA input is needed for this, beyond creation of the relevant nodes. These links display to a seller based on what subtopics have been created underneath the parent node and what groups the seller has permissions for.

## Guidelines for inserting inline links

Inline links are a convenience for users. They can be used to direct a reader toward related information or to provide a path to a destination page where they can perform an action described in the content.

But inline links are difficult to manage when the destination content is deprecated, when titles change, and when a URL needs to be conditionalized differently for different marketplaces. In addition, overuse of inline links can lead to fatigue and confusion for the reader. And, when combined with dynamic links -- such as those generated by subtopic nodes in Help content -- you can end up with awkward double-lists of "See Also" links.

The general rule of thumb with inline links is: Less is more. Avoid an inline link if any of the following is true:

* It will confuse the end user.
* You have already linked to the destination page elsewhere in the topic. If you mention a product page (Example: Manage Inventory) in several places in the same topic, add a link to it only where there is a logical point in the instructions where the seller would need to go to that page to start an action, as opposed to adding the link every time the page is mentioned.
* It requires significant conditionalization but doesn't add much value to the content. Example: "Go to Amazon.com" could be rewritten to "Go to the Amazon home page" and doesn't need a link to be understood, nor conditionalization for other Amazon domains.
* The referenced content is such a small piece of information that repeating the information would not make your current topic overly long; for example, a sentence or two. Users prefer to see all the required information in one topic, rather than being required to navigate to multiple topics, especially with task topics.
* The content you want to reference occurs within the same topic. In this case, use language that refers to the information "found in this topic." Do not use <xref> to set up an anchor tag within the same page or piece of content, or to set up navigational TOCs to sections or subsections within the topic.

Authors should also be mindful of how their links will display if they're reused in other channels. For example, we can use relative links for Seller Central Help content, but URLs have to be spelled out for blurbs and other content stored in CSC.

## Cross references

Use the <xref> tag to create an active hyperlink to the referenced content. For example, use <xref> to link to a related Help page in Seller Central. However, use the <xref> element sparingly. A link in the middle of a sentence could distract the reader and increase the cost of maintaining and translating the topic. It can also limit the reusability of the topic.

When entering a cross reference, include the following information:

Note: Use XMetaL's tools to set up without having to code the following information by hand.

* Use the @href attribute to point to online content. See the following table for the proper syntax.

|  |  |
| --- | --- |
| Situation syntax |  |
| For websites outside Seller Central | http://www.url.com; requires https:// if linking to an Amazon site outside of Seller Central. (Ex. https://www.amazon.com). But do not automatically add the s to websites outside Amazon, as this could break the URL. |
| For links on Seller Central Help pages to other pages published on different nodes in Seller Central | /gp/help/[node ID] |

* Use the @outputclass attribute in the <xref> element to specify the target for your link.

|  |  |
| --- | --- |
| Content being referenced | Output |
| Seller Central Help pages | same\_window |
| Websites outside Seller Central | new\_window; or leave @outputclass blank to default to new\_window |

* To link to an external source, set the @scope attribute to "external". Set links to "external" on all hrefs. The "external" in these cases refers to something external to the CMS. This includes Help topics published on Seller Central. Even though they are stored in the CMS for editing and single-sourcing, the live version of the content that sellers see is delivered (published) "externally" to Seller Central and is separate from the CMS editing and storage tool.

# Chapter 10

Metadata

Metadata classifies the information in a topic or map. You use metadata to find topics in your file system and the CMS. In addition, in HTML output, the presence of search terms in metadata tags increases the ranking of your topic in the search.

You can specify metadata for a topic at three different levels:

* in the topic itself
* at the point the topic is called within a map
* in the content management system

## Map metadata

The <topicmeta> element specifies map metadata about a referenced topic, group of topics, or map. Metadata in a <topicmeta> element applies to all elements contained within the element to which the <topicmeta> is attached. For example, metadata within a <topicmeta> element that appears immediately after the base <map> element applies to all elements in the entire map. However, <topicmeta> attached to a <topicgroup> applies only to the topics within that group, and <topicmeta> attached to a single <topicref> applies only to that topic.

When you include metadata in a topic, it is applied to a specific topic reference, making it less flexible. However, when you include the metadata in a map, you can change the metadata based on the deliverable. For example, if you include information about a group, you might assign different groups in a topic reference depending on the deliverable you are building. But if you include one specific group in your topic-level metadata, you might find it difficult to use that topic in more than one map. In addition, by managing metadata at the map level, you can apply the same metadata to multiple topics without having to enter it multiple times.

Define metadata at the highest possible level to allow for the most flexibility. However, keep in mind the following:

* Metadata specified in <topicmeta> within a map overrides any metadata values for the same elements specified in the topics. Therefore, only use <topicmeta> for customer-facing metadata.
* Metadata specified within a map is not cascaded to the individual topics until runtime. Any metadata that will assist authors in locating topics in the CMS should remain in the topics themselves.
* You can use all of the same elements within <topicmeta> as within the <metadata> tag in the Prolog of a topic, except index terms. However, <keywords> does not need to be nested within a <metadata> tag (although it could be). See Adding metadata for more information.

## CMS metadata

Authors can use CMS metadata to search for content in the repository. CMS metadata can also be used to run queries and reports. Our CMS automatically captures the following metadata for all content stored in it:

* Author name
* Creation date
* Last-modified by, name and date
* Workflow stage; for example, Draft, Prerelease, Released
* Information type (task, concept, reference)
* Version numbers
* GUID
* Revision information

The CMS also stores optional metadata such as:

* Description: We use this to identify what countries have conditions in a topic
* Project name: We use this for translation management
* Translation pipeline
* Translation billing name

# Chapter 11

Attributes

Attributes specify properties of the content that can be used to determine how the content should be processed. We use them to:

* Support content referencing
* Support translation and localization
* Display content in specific ways

Many attributes are mentioned in this model within the context of the elements on which they can be used. However, this section provides a summary of common attributes that can be used on many different elements.

### ID attribute

The @id attribute assigns an identifier to an element so the element can be referenced and reused. It is required on all root elements. All other elements can have an @id attribute, but it is not required unless you plan to reference the element. All topic IDs must be unique, and IDs must be unique across all elements within a single topic. If two elements have the same ID, all references will go to the first element with the ID. For this reason, the CMS is configured to automatically assign an ID to topics and key elements. Automatic IDs will guarantee all topic IDs are unique.

**Note:** IDs do not need to be unique across topics. Since the topic @id is also part of any reference made to an element, the reference will be unique if elements in different topics have the same @id.

The CMS and XMetaL are configured to automatically assign IDs to the following elements each time you include the element in a file:

* <map>
* <concept>
* <reference>
* <task>
* <section>
* <example>
* <table>
* <note>
* <step>
* <ol>
* <ul>
* <li>
* <uicontrol>
* <title> (topic)

Optionally, you can provide your own ID to make it more meaningful when you are looking for the appropriate item to reference. When creating your own ID, follow these guidelines:

* IDs should be unique across all elements within a topic. If two elements have the same @id, all references will go to the first element with the ID.
* Start the @id with a letter. Thereafter, use only alphanumeric characters and the punctuation marks underscore (\_), hyphen (-), period (.), and colon (:). No other characters including spaces are allowed.
* Start the @id with an indication of the type of element being referenced (for example li\_ or step\_).
* When referencing elements that have a title, such as sections and tables, use the title as the ID, removing spaces between the words and using camel case (capitalizing the first letter of each word). There is no limit to the length of the @id.
* Note that @id's for some elements do not always pass through the transform into published code.

### Display attributes

Display attributes affect the way in which an element is rendered in your final output.

| Name | Description | Available Values | Required |
| --- | --- | --- | --- |
| @outputclass | Specifies that a special styling be used during output processing. | notitle - on topic <title>, suppresses title publication  hbtitle - on topic <title>, publishes title as a module subheading. Use only on pages with multiple modules that will display together, and only if you know the topic is not being reused elsewhere.  same\_window - on <xref>, allows end user to open link in the same browser tab or window. Use for Seller Central pages.  new\_window - on <xref>, allows end user to open link in a new browser tab or window. Use for pages that are external to Seller Central. However, leaving this field blank will default the UX to a new window.  strike - on <ph>, publishes strikethrough text  code - on <ph> or <p>, publishes as monospaced font with a yellow shaded background; intended for publishing short snippets of code  passthrough - on <codeph>, allows HTML or Javascript code to pass through with no transform. Only usable with HB 4.0 transform, which does not have the Batch Publishing feature. Must get permission from the IA or a CMS admin to use.  buttonImage - on <xref>, sets up automatic "button" formatting within text; set the <xref> @scope to external. See Appendix A for details on creating inline buttons. | No |

### Localization attributes

The @translate and @xml:lang attributes identify language-specific words or phrases for specific processing (or nonprocessing, in the case of translate="no").

| Name | Description | Available Values | Required |
| --- | --- | --- | --- |
| @translate | Indicates whether the content of the element should be translated or not. | yes | no | Only to change from the default of yes. |
| @xml:lang | Specifies the language of the topic content. Set the @xml:lang attribute to "en-US" for every topic at its root. Do not set this attribute for individual elements. | en-US | Yes, on the root element of each topic. The CMS will automatically assign this attribute when you create a new source topic, defaulting to en-US. |

# Chapter 12

Reuse Strategy

With careful planning and conditional processing, we can reuse single-source information among marketplaces.

|  |  |
| --- | --- |
| If the following is true | Use this approach |
| Topics can be used without any changes in another output or publication. | Reuse topics with conrefs. |
| You want to reuse portions of another topic within your current topic. | Reuse elements through libraries. |
| Indicating destination marketplaces for content elements in a single-source topic. | Filter content with conditional processing based on specific data (such as destination country). |

## Writing for reuse

Before reusing content, carefully analyze whether the content can be used exactly as it already exists, or whether it must change because of real differences. Often, multiple topics are created that say essentially the same thing because of personal preferences among authors, editors, and subject matter experts. But in a reuse-centric environment, these preferences must be set aside to gain the maximum reuse benefit.

To avoid creating repetitious content when an existing topic can be reused, authors should follow these guidelines when creating new content:

**Establish a "single source of truth."**

For reuse to work, authors need to accept that common information need be written in one and only one topic. Even if the way it is written is not the way an author might have written it, it is a waste of time and budget to rewrite the same information. Further, propagating different versions of the same information increases the likelihood that not all versions will be updated in the future. In addition, multiple explanations of the same content can be confusing to end users.

**Search for an existing topic before writing a new one.**

Before creating new content, search the CMS to find content that has already been written and reuse it. To help with your search, use metadata, keywords, phrases, or other criteria to find content that is reusable for your assignment. Searching for reusable content can save hours of writing time and reduce repetitious content within the repository.

**Write clear, descriptive titles.**

Topic titles are one of the most important cues to your end users in helping them find information. The more descriptive the title, in words that are meaningful to the end user, the easier it is for users to read and comprehend the information. Writing meaningful titles helps set the context for the information and frequently lessens the need for explanatory paragraphs within the topic. Clear, descriptive titles also help other writers find content that might be appropriate for them to reuse.

**Include thorough metadata.**

Metadata, describing the content and the situations in which the content is applicable, is just as important as the actual content in a reuse environment. If others don't know what content a topic contains or they can't find it, they can't reuse it.

**Ensure topics are stand-alone and cover one complete subject only.**

Well-constructed topics cover a single subject and do not require any other topics to be understood by the end user. When you mix content types in a single topic or include multiple subjects, you limit the situations in which the topic might be reused. All subjects covered by the topic must apply in any situation in which you want to reuse it. By separating subjects and content type, you have more flexibility in which information you reuse.

**When details are not important, leave them out.**

The more explicit the details you provide, the more difficult to reuse that content in other situations where the details may not all apply. Use all-compassing words and phrases like the following that indicate details are not complete or that all details may not apply in certain situations:

* such as
* including
* where applicable
* Use generic terms whenever possible and variables when you must be specific.

## Reusing topics

The simplest form of reuse is just using topics exactly as they were created the first time in a new output. For example:

* The same reference topic might need to appear on multiple pages for different sets of sellers.
* A topic could apply to multiple audiences or features in different output types. For example, although specific requirements do differ across various services, all users might require background information on the same conceptual information before they begin learning the specific details of their service. Such concept topics can be written once and referenced from different outputs.

If a topic cannot be reused exactly as is, it's possible you might be able to make revisions to extend its use. Be sure to check where else the topic is used and consider the implications of your change in these other contexts. If the changes do not apply in all situations, consider how you might implement other reuse mechanisms to accommodate every situation.

## Reusing elements through libraries

Libraries (collection files) hold content that you want to reuse by reference in other topics. They enable you to edit similar content in one location and maintain simple and direct links between small pieces of reusable content and the topics in which they are used. By using libraries, you avoid complex inter-topic linking structures that are difficult to maintain and that rely on authors knowing which specific, individual topics contain information that might be useful to them. Instead, authors are able to easily locate reusable content in common folders and files. In addition, creating libraries helps to protect shared content from changes that would not apply in all reuse situations.

When making changes in libraries, authors are aware that the content is reused and that they need to be aware of implications for changing the content. For example, translations of topics that contain content referenced from a library cannot be published unless the library is also translated for the same languages.

### When to use

Use libraries any time that you want to reuse portions of another topic within your current topic. Do not reuse content from a file that is not a library. Instead, move common content to a library and then reference it by inserting a content reference (conref) into your topic.

Note: Content stored in a library should not be conditionalized in the library itself. Conref the content into a <p> or <section> element within your current topic, and conditionalize it there.

It is possible to create a topic body that is nothing but content references to other topics. However, if you find that all the references go to the same topic, you should probably be simply reusing that topic instead of creating a new one. In addition, in all cases, the topic title and short description must be unique. Do not conref titles or short descriptions.

### Creating and structuring libraries

Move shared content into libraries on the first instance of reuse and point both the original and new topic to the library. Do not reuse content from a file that is not a library. This practice reduces the overhead of managing and controlling common content.

Do not create new libraries without permission from our Information Architect or a CMS admin.

When creating or adding information to a library, follow these guidelines:

* With the obvious exception of elements exclusive to task topics, use <concept> or <reference> topics for your libraries. Use elements such as <section> or <note> to logically sort the collected content into related groupings so authors can more easily find information.
* Because task topics will not allow you to nest their unique elements within a <section> element, you will have to rely on XML comments to locate the information within the collection.
* By storing reusable information by element type, authors can more easily locate information when it is needed. Sort content into element-specific libraries, such as:
  + Notes
  + Warnings
  + Cautions
  + Steps
  + Prerequisites
  + Postrequisites
  + Descriptive paragraphs

Note: Do not create libraries for <title> or <shortdesc> elements. These elements should be unique for each topic to help distinguish it from other topics.

* Libraries are stored in the common Libraries folder.
* Include content no smaller than a sentence in length.
* Ensure each element in the file has a unique ID. Use XMetaL to generate an ID if it does not automatically assign it when you add the element.
* Keep in mind that any nested elements are included when you reference an element. Be sure to nest all content that will be used together in a single element. However, if nested content might also be reused alone, ensure that you have provided IDs on those nested elements as well. If an element might be reused without its nested content, do not nest the content in the library; in situations where all content is needed, you will have to include multiple references.
* Use XML comments to document your libraries. Include information that helps identify where each element in the file is referenced as well as metadata keywords that might help authors find the specific content they need.

## Conditions

Conditional processing allows authors to label content according the situations to which it applies. When you later process the content, you can use these labels to exclude content that does not apply to the current situation. This enables you to manage content for multiple purposes in a single source.

### How to use

We use conditions to label content according to marketplace. All content in a single-source topic must be conditionalized by country to indicate where it will eventually publish.

When you are revising or reworking content that has been previously conditionalized, do not accept the existing conditions at face value. Look for ways we can streamline and simplify the text, such as making an wording fix in the source content that eliminates the need for a localized term. Strive for parallel organization wherever possible so that, for example, we don't set the same text as a <p> for some countries and a <note> for others.

Other best practices:

* Enter all conditions in the single-source topic file, which is stored on the en-US layer in the CMS. The en-US label does not mean "for the US marketplace." Rather, it indicates the language the content starts from: global English.
* All conditions are written in global English in the single-source topic, regardless of where they originate or where they will ultimately publish. The single-source topic is later translated to facilitate native languages for each OU and for language-switcher features.
* Create new content with conditions only when the meaning of a sentence or paragraph is different from the global English. For example, a paragraph that contains dollar signs for the US marketplace could remain the same for Canada if the values were equivalent for both marketplaces. However, for the UK, the dollar signs would need to be replaced with a pound sign. For Germany, France, Italy, and Spain, a euro sign would be required.
* Do not create a new set of conditions if the only differences in the text are linguistic preferences, such as moving punctuation outside of quotation marks.
* Apply country conditions to all content, even if the condition will be published with the same phrasing for all marketplaces.
* Target <p>, <ul>, <note>, <dlentry>, and <step> elements whenever possible. Only add conditions to a <section> element when it contains one element (plus a title).
* Do not set conditions on inline elements that call out a sentence, term, or word. For example, if a paragraph has three sentences and only one is different, you should set a condition on the entire paragraph at the <p> element.
* For tables, conditionalize at the <table> level and not row by row or cell by cell.

**Exceptions:** There might be exceptions where large tables are identical for multiple OUs with maybe one or two cells containing localized content, such as an example of a phone number or address. In that case, it is acceptable to use a <ph> element to conditionalize the content of the cell, but not the cell itself.

* For bulleted and ordered lists, conditionalize the entire list as a block at the <ol> or <ul> level rather than on individual <li> elements.
* Apply <ph> elements to conditionalize section titles. This is one of the rare exceptions for using a <ph> element to conditionalize. If you try to conditionalize a <title> element in a section without the <ph>, the Set Condition menu option will be grayed out in XMetaL.
* Set blocks of related conditionalized text near each other rather than at the end of a section or page, or grouping unique content by country. In the source file, this sometimes looks like several near-identical paragraphs in a row, but it will make it easier for future editors to interpret.
* Avoid nested conditions. That is, do not set conditions on a parent element (ex., <ul>) and then set different conditions on a child element (ex., <li>). It is easy to set these up incorrectly and accidentally exclude content. Also, some elements do not support nesting.

**Exception 1:** If you are conditionalizing individual vertical lists, you can conditionalize the <ul> element and then the <li> elements that are different. This eliminates the need to add conditions to every list element. DO THIS SPARINGLY. If you need to add conditions to a lot of bullets to make this work, do not do this.

<ul (US or UK or DE or FR)>

<li>Bullet 1... </li>

<li>Bullet 2...</li>

<li (US)>Bullet 3...</li>

<li (UK or DE or FR)>Bullet 3...</li>

</ul>

**Exception 2:** If you are adding a <ph> element to a table cell for conditions, also add conditions to the <table> element. This eliminates the need to add conditions to every cell.

Note: If a condition is set for a nested element (the <li> or the <ph> element in the exceptions above), but is not set for the containing element (the <ul> or <table> element), the nested element will never publish. All nested conditions must be included in the containing element.

* When possible, adjust the source text to avoid the need for conditions altogether. Example:

|  |  |  |
| --- | --- | --- |
| US Content | DE Content | Resolution: Edit source text |
| Amazon.com is always looking for ways to help you grow your business. | Amazon.de is always looking for ways to help you grow your business. | ​Amazon is always looking for ways to help you grow your business. |

* Use <draft-comment> to document what has been done in the file so another author can easily interpret it.

### Rules for condition output

Before setting conditions, make sure you fully understand the following rules for conditionalization output:

* Content that is not conditionalized for a given condition will be included for all values of that condition. Authors add conditions to all elements to avoid the risk of "universal" content accidentally publishing in a marketplace where it is inaccurate.
* Content with conditional values that match the given condition value will be included in the processing. For example, any content labeled with the condition country=France will publish when you select FRAmazon as a destination marketplace in Ajuda.
* Content with conditional values that do not match the given condition value will be excluded from the processing. For example, content labeled country=Spain will not publish when you select INAmazon as a destination marketplace in Ajuda.
* You can specify more than one value for any conditional attribute. Content that matches any of the values will be included. For example, you would set a condition such as *country=Canada or country=France* on a <p> element that is common to both OUs.
* You can specify more than one condition on the same element. Content that matches all of the values will be included. We use this approach only in rare cases, such as when indicating language-specific content like videos.

<p country='United States'>Download your file template.</p>

<p country='United States' and language\_code=en-US><xref>Template in English</xref></p>

<p country='United States' and language\_code=zh-CN><xref>Template in Chinese</xref></p>

# Chapter 13

File management

## Folder structure

Within the Seller Support branch of the CMS, folders at the highest level are organized by file type: publications, maps, topics, images, library.

Subfolders can be used within the main folders for additional organization. However, we do not currently organize files via subfolder. Authors are encouraged to use repository search functions to find objects stored in the CMS. Generally, the quickest way to find Help topics is by entering the parent and module IDs in the Help Builder and Blurb Metadata facet on the Search tab of a repository window.

The CMS allows authors to create multiple objects with the same topic metadata, such as parent ID. To help identify which object is "active," look for CMS metadata such as:

* **Folder and subfolder location.** Almost all of our content is stored in the SOA Help - Production subfolder. Active objects found outside this location should be moved into this subfolder. Objects found in the SOA Help - Deprecated subfolder can usually be safely ignored.
* **Last-modified date.** Be aware, however, that this date can change even if a user only Views the topic. So it is not an indication that an actual revision was made.
* **Description.** Content that has been single-sourced and conditionalized (a process that began in 2016) will have information such as "Conditions: US, CA, UK, DE, FR, IT, ES, JP, IN, MX, CN, BR" in this field. The Description might also include other helpful notes such as "Deprecated - Do not use."

To keep the CMS well-organized, folders and subfolders should only be created by our Information Architect or CMS administrators.

## File naming conventions

* Topic titles: Use the content in the topic file's <title>

# Chapter 14

Special use cases

## Inserting "FAQs" in long-form Help

**Note:** The "FAQs" discussed in this section are not to be confused with the FAQ deliverable for the SOP team, or the FAQs used by the Athena team to provide actionable snippets of information.

It is not uncommon for us to get requests to place a list of FAQs into long-form Help content.

FAQs in draft content and BRDs can help us identify potential blurbs. But we discourage inserting them into long-form Help content. This is because if a question is indeed being asked frequently, it usually signals an information gap elsewhere in the UI or in the Help content that needs to be filled to fix the experience for end users.

Our preferred approach, therefore, is to either fix the UI or work the FAQ answers elsewhere into related new or existing content.

If you identify a use case where you feel FAQs are the most user-friendly way to provide certain information, follow these guidelines:

* Keep the FAQs short. Merge related questions and answers together. If several questions have the same answer, group these together rather than duplicating the answer multiple times.
* Be mindful of the overall length of your Help page. The content in the FAQs should not repeat content covered elsewhere on the page.
* Use <dl> to format the questions and answers. If you come across FAQs in an existing topic that were set up as <section> elements, convert these into a <dl>.

<dl><dlentry><dt>FAQ question text</dt>

<dd>FAQ answer text</dd>

</dlentry></dl>

# Appendix A

Element Quick Reference

| I want to enter: | DITA tag | Example | Notes | For more information |
| --- | --- | --- | --- | --- |
| Acronym | Do not tag. | <p>Enter your PIN.</p> |  |  |
| Bold text | Use appropriate semantic tag for type of content.  Use <b> for cases of emphasis only | <p>Do <b>not</b> use this element if there is a semantic tag that is appropriate.</p> | Examples of semantic tags include <wintitle> and <uicontrol>. The stylesheet will format these with emphasis. | Using Inline Elements |
| Book or magazine title | <cite> | <p>Refer to the <cite>Chicago Manual of Style</cite> for guidelines on comma usage.</p> | Formats in Seller Central as italic. | Using Inline Elements |
| Bulleted list | <ul> | <p>There are five types of lists in DITA:<p>  <ul> <li>ordered</li> <li>unordered</li> <li>simple</li> <li>definition</li> <li>parameter</li>  </ul> | Do not embed list elements within <p> elements. | Creating Lists |
| Button graphic, inline (Seller Central Help only) | <xref>  <ph> | <p><xref><ph><ph>Request approval</ph></ph></xref></p> | For <xref>: Set @outputclass as buttonImage. Set @scope to external. Set @href to the URL where the button should direct.  For the first <ph>: Set @outputclass as awesomeButton buttonLarge primaryLargeButton inner\_button  For the second <ph>: Set @outputclass as button\_label |  |
| Button name (UI label) | <uicontrol> | <p>Click <uicontrol>Advanced </uicontrol>.</p> | The stylesheet formats these with bold emphasis. | Using Inline Elements |
| Centered text | Within a table cell, set @align to center.  Do not tag body text. | <table> <tbody><row> <entry>Backlight</entry> <entry align="center">X </entry> </row></tbody>  </table> | Use Table Properties to center text within a cell. However, this doesn't always render as intended in the output.  Placement of body text on the page is controlled through the stylesheet. You cannot force text to be centered. |  |
| Code | <ph>  <p>  <codeblock> | <p>Use <ph>SELECT\*</ph> to choose all customers.</p>  <codeblock>// my first program in C++ #include <iostream> using namespace std; int main ()  {count  << "Hello World!" ; return 0; }  </codeblock> | Use <ph> for inline code snippets or <p> for text blocks where line breaks aren't needed. Set @outputclass as "code".  Use <codeblock> for multiple lines; line breaks are preserved in these cases. Formats in a monospaced font. | Using Inline Elements |
| Column heading reference | <uicontrol> | <p>Refer to the <uicontrol>Name</uicontrol> column.</p> |  | Using Inline Elements |
| Command name | <uicontrol> | <p>Use the <uicontrol>Start </uicontrol> command to...</p> |  | Using Inline Elements |
| Comment | <draft-comment> | <draft-comment>**Draft Comment:** Did you mean to make these revisions in the text?  </draft-comment> | Use the Attribute Inspector to add an @author and a @time, to help future editors know who made the comment and when. Delete all draft comments after they're addressed. | Using Inline Elements |
| Component or feature name | Do not tag. | <p>The Spellcheck function ...<p> |  |  |
| Cross reference | <xref> | <p>For more information, visit <xref>Managing Orders</xref>.</p> | Set the destination in @href. Do not use <xref> to try to create content anchors within the same topic. | Cross References |
| Definition in text | <term> | <p>A <term>verb</term> is a word used to describe an action, state, or occurrence.</p> | Because of possible reuse, you won't know where the term appears in the sequence of all possible locations the topic is used. Do not think of this as a "first occurrence" strategy.  If you use <term>, include it when the definition is part of the same sentence.  Formats as italic in Seller Central. | Using Inline Elements |
| Definition list | <dl> | <dl> <dlentry> <dt>DITA</dt> <dd>Darwin Information Typing Architecture</dd> </dlentry>  <dlentry> <dt>XML</dt> <dd>eXtensible Markup Language</dd> </dlentry>  </dl> | Formats in Seller Central as a bold H4 title with text in regular paragraphs. | Creating Lists |
| Dialog box name | <uicontrol> | <p>In the <uicontrol>Save As </uicontrol> dialog box, select the new file type.</p> | Formats with bold emphasis in Seller Central. | Using Inline Elements |
| Email address | <xref> | <p>If you have questions, email <xref>contact-us@amazon.com</xref>.</p> | Use a mailto link in the @href field in the Attribute Inspector. Set the scope to external. |  |
| Emphasized text | <b> | <p>Do <b>not</b> use this element if a semantic tag exists.<p> | Do not use when a semantic tag exists. | Using Inline Elements |
| Example text with formatting | <codeblock>  <lines> | <p>The following is an example haiku: <codeblock>Falling to the ground, I watch a leaf settle down In a bed of brown.</codeblock>  </p> |  | Using Inline Elements |
| Field name | <uicontrol> | <p>Type the full path to your source file in the <uicontrol>Source File</uicontrol> field. <p> | Formats with bold emphasis in Seller Central. | Using Inline Elements |
| Footnote | <fn> | <p>Your suitcase can weigh no more than 50 pounds<fn>20 kg</fn>.</p> | Places the footnote at the bottom of the topic when published. | Using Inline Elements |
| Hardware label or button name | Do not tag. | <p>Press ENTER on your keyboard.</p> |  |  |
| Hyperlink to Seller Central page | <xref> | <p>For more information, visit <xref>Selling on Amazon Schedule of Fees.</xref></p> | Enter the URL in the @href field in the Attribute Inspector, and make the link relative. Set the scope to external. Set @outputclass to same\_window. |  |
| Image | <image> | <p>Click <image>[placeholder or preview]</image>.</p> | Use the @href field to enter the URL of images stored outside the CMS. Set the scope to external. Images with URLs are signified by a placeholder X icon in XMetaL, but the image will publish on output.  Set @placement in the Attribute Inspector to inline, for inline images such as icons, or to break if the image needs to be on its own line. | Images and Media Content |
| Indented or offset text | Do not tag. |  | Placement of body text on the page is controlled through the stylesheet. You cannot force text location within DITA. |  |
| Italics | <i> | <p>Do <i>not</i> use this tag if a semantic element exists.</p> | Do not use this tag for emphasis if an appropriate semantic element exists. | Using Inline Elements |
| Internet address | <xref> for active hyperlinks. Otherwise, do not tag. | <p>For more information, visit <xref>www.services.amazon.com.</xref></p> | Enter the URL in the @href field in the Attribute Inspector. Set the scope to external. |  |
| Keyboard key | Do not tag. | <p>Press ENTER.</p> |  |  |
| Keyword | Do not tag in text. Use a <keyword> entry in the topic <prolog> instead. | <prolog> <metadata> <keywords><keyword>information model</keyword><keywords> </metadata>  </prolog> |  | Adding metadata |
| Line break | Do not tag for individual line breaks.  To preserve line breaks for specific types of content, use <lines> for body text or <codeblock> for code. | <lines>Shall I compare thee to a summer's day?  Thou are more lovely and more temperate.</lines> |  |  |
| Menu selections | <uicontrol> | <p>Under <uicontrol>Inventory </uicontrol> select <uicontrol> Manage Inventory</uicontrol>.</p> | Formats in Seller Central as bold. | Using Inline Elements |
| Monospaced text | Tag with the appropriate semantic tag for the type of content.  For example, <codeblock>. |  | The stylesheet will format certain semantic tags in a monospace font. | Using Inline Elements |
| Non-translatable text | <ph> | <p>Inventory File Templates in Simplified Chinese <ph>(库存文件模板)</ph></p> | Set @translate to no in the Attribute Inspector. |  |
| Note | <note> | <note>**Note:** Be sure to apply your changes before leaving this tab.</note> | "Note" is the default lead-in if the @type field in the Attribute inspector is blank. The lead-in is inserted automatically when you place the element. | Writing Notes |
| Numbered list | <ol> | <p>The five most populous cities in the United States are:  </p>  <ol> <li>New York, New York</li> <li>Los Angeles, California</li> <li>Chicago, Illinois</li> <li>Houston, Texas<li> <li>Philadelphia,Pennsylvania</li>  </ol> | Do not embed list elements within <p> elements. | Creating Lists |
| Program name | Do not tag. | <p>The Microsoft Office suite includes Word.</p> |  |  |
| Section with subheading | <section> | <section><title>My awesome section</title> <p>This section will have a subheading on the page.</p>  </section> | When the content warrants it, create a new DITA topic rather than using <section>. |  |
| Special character | Use XMetaL's Insert > Symbol command.  You can also type unicode and hexadecimal codes directly in Plain Text View. | <p>Sell your item for 1¢.<p> |  | Special Characters |
| Steps | <steps>  <step>  <cmd> | <steps> <step><cmd>Select <uicontrol>File</uicontrol> and then <uicontrol>Save</uicontrol></cmd></step> <step><cmd>Browse to the location where you want to save the file.</cmd></step>  </steps> | Steps must be entered in a Task topic type. Do not use an <ol>. | Task Elements |
| Strikethrough text | <ph> | <p>The fee is <ph>$10</ph> $20.</p> | Set @outputclass to strike. |  |
| Subheading | <title> inside a <section> | <section><title>Unordered lists</title> <p>Unordered lists...</p>  </section> | You cannot create sections within task topics.  You can have only one <title> per section. If you insert multiple <title> elements, only the first one will publish on the page. | Concept Elements, Reference Elements |
| Subscript | <sub> | The chemical symbol for water is  H<sub>2</sub>O. |  | Using Inline Elements |
| Substeps | <substeps>  <substep>  <cmd> | <step><cmd>Attach the tires.</cmd> <substeps>  <substep><cmd>Stretch the deflated tires over the metallic rims of your bike.</cmd></substep>  <substep><cmd>Insert the air valves through the circular openings in the wheel frames.</cmd></substep>  <substep><cmd>Inflate each tire. </cmd></substep>  </step> |  | Task Elements |
| Superscript | <sup> | <p>A general quadratic equation takes the form ax<sup>2</sup>+bx+c=0.</p> |  | Using Inline Elements |
| Tab name (UI label) | <wintitle> | <p>On the <wintitle>Manage Alerts</wintitle> tab, ...</p> | Formats with bold emphasis in Seller Central. | Using Inline Elements |
| Table | <table> | <table> <tgroup> <title>Troubleshooting</title><thead> <row><entry>Symptom</entry><entry>Cause</entry><entry>Solution</entry></row> </thead> <tbody> <row> <entry/> <entry/> <entry/> </row> </tbody> </tgroup>  </table> | Use XMetaL's Table commands to insert and work with tables. | Building Tables |
| Tip | <note> | <note>**Tip:** To improve computer  performance, invest in a high-quality registry cleanup program.  </note> | Set @type to "tip" in the Attribute inspector. It will adjust the tag code and insert the Tip lead-in automatically. | Writing Notes |
| Trademark | <tm> | <tm>Microsoft Excel</tm> | @tmtype is a required attribute. Enter reg for a registry mark. Enter tm for a trademark. | Using Inline Elements |
| Underlined text | <u> | <p>Do <u>not</u> use this tag solely for formatting purposes.</p> | Formatting of content is controlled by the stylesheet. Code the content with the appropriate semantic tag. |  |
| URL | <xref> for active hyperlinks. Otherwise, do not tag. | <p>For more information, go to <xref>www.services.amazon.com</xref>.</p> | Enter the URL in the @href field in the Attribute Inspector. Set the scope to external. |  |
| User interface control | <uicontrol> | <p>Click on the <uicontrol>Save</  uicontrol> button.</p> | Use for anything clickable or selectable in the user interface, such as menu options or buttons. Also use for labels that appear as UI elements, such as column headings in reports. Formats with bold emphasis in Seller Central. | Using Inline Elements |
| Video | <object> | <p><object data="sample.swf"> [Placeholder image]</object></p> | Insert video objects inside a <p> or <section>. Set the @classid to sellertutorial. If you are linking to a video stored outside the CMS (such as YouTube), insert the URL into the @data field. You can also use the Attribute Inspector to set height and width of the video player. Use only numbers (no units) for height and width. Ex. 480, instead of 480 px. | Adding Videos and Animations |
| Warning | <note> | <note>**Warning:** Follow the lockdown procedure before servicing.</note> | Set @type as "warning" in the Attribute inspector. It will insert the Warning lead-in automatically. | Writing Notes |
| Web address | <xref> for active hyperlinks. Otherwise, do not tag. | <p>For more information, go to <xref>www.services.amazon.com </xref>.</p> | Enter the URL in the @href field in the Attribute Inspector. Set the scope to external. |  |
| Window title | <wintitle> | <p>You can also set your margins in the <wintitle>Print Preview</  wintitle> view.</p> | Use for titles of Seller Central feature and content pages. Formats as bold. | Using Inline Elements |

# Appendix B

## Sample Concept (unconditionalized)

<concept id="GUID-6891C514-8CBA-4306-940B-A585A9F07614" xml:lang="en-US">

<title id="GUID-97B53FD5-32EB-4ACD-BDE2-5FC35C5AD912" outputclass="notitle">Increase Your Chances of Winning the Buy Box</title>

<prolog id="GUID-94DC661A-4EBE-4C75-96F9-4813AC106839">

<metadata>

<keywords>

<keyword id="GUID-D9C6D6C2-5F70-4CDB-BF57-2D48E93D4007">win buy box</keyword>

</keywords>

<othermeta name="parentId" content="201687570"/>

<othermeta name="topicId" content="201687580"/>

<othermeta name="language" content="en\_US"/>

<othermeta name="country" content="US"/>

<othermeta name="sortIndex" content="1"/>

<othermeta name="group" content="2011"/>

</metadata>

</prolog>

<conbody id="GUID-173CE293-3B47-4D76-A6AA-500759EAB1D2">

<section id="SECTION\_5F0A7A33105042D6868C671C39FF103C">

<p>Since seller performance metric targets can vary by category and are subject to change, we do not disclose specific targets for becoming Buy Box eligible. Buy Box eligibility requires meeting very high standards. Excelling in all areas that deliver an outstanding customer experience is the best way to achieve Buy Box eligibility and win the Buy Box, including the following:</p>

<p><uicontrol id="GUID-2534047E-ECEF-4D36-88E1-3BCE9C697FF2">Price</uicontrol></p>

<ul id="UL\_EF872EB9611E492A95C7E209397FAFDA">

<li id="LI\_FB2DCD09E98242AC99F279C8AD4A9439">Price competitively. Customers trust that they will find low prices on Amazon.</li>

<li id="LI\_4946F30FADB54F42A92F0331B47B71BC">Watch your competition to stay competitive. View the price of the offer that is currently winning the Buy Box. On the Manage Inventory page, click the Preferences button, and select Show when available for the current Buy Box winning price. Buy Box winning prices are displayed for "New" condition listings only. </li>

</ul>

<p><uicontrol id="GUID-1FC45EEA-5645-43CC-BAEB-D1068BD6B3A6">Availability</uicontrol></p>

<ul id="UL\_377FF739F3C240D2B2CBEE547F5B0011">

<li id="LI\_B5555056B28E4FBAAF6EEE446B05D9FD">Keep stock available. If you have no current stock for a product, you cannot win the Buy Box. Use inventory planning best practices to keep your popular products in stock.</li>

<li id="LI\_C68783634D1145AE86F0FFB115A1145D">Update your inventory. Customers who submit orders for products that are not in stock are likely to leave negative feedback, especially around the holidays. To ensure that buyers cannot place orders for out-of-stock items, keep your inventory updated.</li>

<li id="LI\_F20FB1DE32724578BE85C4D9960986CD">Plan vacations. If you are going on vacation, make sure that customers are not left waiting for orders to arrive. See <uicontrol id="GUID-2C8BBDBB-338F-41E8-999A-7E6A8BEC53A0">Listings Status</uicontrol> for more information. </li>

</ul>

<p><uicontrol id="GUID-2B71E07F-F523-4862-8F07-547EC79B8970">Fulfillment</uicontrol></p>

<ul id="UL\_4F510925B3A94283BB7B2C8343D2A86C">

<li id="LI\_9670C95FA3A14113A2F8D8CEFFF83746">Offer multiple shipping options on your products, including free shipping. You can manage your available shipping options in <uicontrol id="GUID-5A5E2B3E-0131-4FE9-B6F3-676C5F224916">Shipping Settings</uicontrol>.</li>

</ul>

<p><uicontrol id="GUID-0708A579-05D1-4ADB-B943-828376A5EB84">Customer Service</uicontrol></p>

<ul id="UL\_999EC862D8904383B652ED8E4A5BB662">

<li id="LI\_365215F625B7479881EAD98312574070">Customer service is measured in numerous ways. Excelling in all areas can help you succeed on Amazon. The <wintitle id="GUID-BC5BBDC4-E87E-4B94-BD2A-CF7A9F2A9682">Account Health</wintitle> page on the Performance link can help you track your customer metrics. </li>

</ul>

</section>

<section id="SECTION\_7D8B016B2B5E45FDA7A4163D6A5C6B90">

<title id="GUID-B36628B2-3FB0-4BA5-AFBC-8BD14912066F">See Also</title>

<ul id="UL\_DCD01FFE69304DEF8EA57A436144FF0C">

<li id="LI\_6153E44EF46147118EAA2F36C9A9EF36"><xref href="/gp/help/17191" outputclass="same\_window" scope="external">The Power of Free Shipping</xref></li>

<li id="LI\_96E645D2D9C444AB801D5174A707C734"> <xref href="/gp/help/200135620" outputclass="same\_window" scope="external">Listing Status Settings for Vacations, Holidays, and Other Absences</xref></li>

<li id="LI\_3DF88A9AE8A34794A9954C10A1FB4551"><xref href="/gp/shipping/dispatch.html" outputclass="same\_window" scope="external">Current Shipping Rates and Settings</xref> </li>

<li id="LI\_F2D7B41727F74FD3957541928041A490"> <xref href="/gp/seller-rating/pages/account-health.html" scope="external" format="html" outputclass="same\_window">Account Health</xref></li>

<li id="LI\_A6FE3CC09CCD4C7B809A13A605E34E2C"><xref href="/gp/help/200370550" outputclass="same\_window" scope="external">Seller Performance Measurement</xref>.</li>

</ul>

</section>

</conbody>

</concept>

## Sample Reference (unconditionalized)

<reference id="GUID-12D9EC96-E02B-4502-993D-F952E6D0E07F" xml:lang="en-US">

<title id="GUID-2C1DF364-8FA3-4387-BCDB-2A63B7C51B64" outputclass="notitle">AmazonOrderItemCode</title>

<prolog id="GUID-D027CA83-DBE4-4DE2-94C1-FAEB4B33AADB">

<metadata>

<othermeta name="parentId" content="37731"/>

<othermeta name="topicId" content="200088630"/>

<othermeta name="language" content="en\_US"/>

<othermeta name="country" content="US"/>

<othermeta name="sortIndex" content="1"/>

<othermeta name="group" content="1671"/>

<othermeta name="group" content="3121"/>

<othermeta name="group" content="4601"/>

<othermeta name="group" content="4661"/>

<othermeta name="group" content="6161"/> </metadata>

</prolog>

<refbody id="GUID-92B5A89C-FC8F-4FFF-8FBA-1E47514241BF">

<section id="GUID-23032991-0A9F-4AB1-8338-7020D768D6BE">

<title id="GUID-73D0261F-2B57-4240-8A3F-66A14816728C">Element</title>

<p>Amazon's unique, displayable identifier for an item within an order</p>

</section>

<section id="SECTION\_C44EFE84EA0F4F84AF224B126A211E1A">

<title id="GUID-D9197509-0142-449F-AFBD-AE1BFEAD49FB">Requirements</title>

<ul id="UL\_3CBEB0D4AB5F4B1387F03A38C8FC12F7">

<li id="LI\_996D25419EDE4144A7CEF0C6950050F8">Required: Yes</li>

<li id="LI\_E8450AB817AD493DA11DFE5A217E4CCC">Type: String, 14 alphanumeric characters</li>

<li id="LI\_2C9B089D2880426C8EA07448823F7F25">Default: none</li>

</ul>

</section>

<section id="SECTION\_808892DBC42E40318EE83DF42F740C1E">

<title id="GUID-233CC108-A49B-4890-8CFC-08F12F49C2EB">Special Notes</title>

<p>Identifier for an item within an order. Used to identify an item within an order. The AmazonOrderItemCode applies to a specific order - do not use an AmazonOrderItemCode in more than one order.</p>

</section>

<section id="SECTION\_F66B0CC02C564363B9F0227B5B0355C0">

<title id="GUID-9ABF7E91-7451-4545-ABD8-F365A7615A05">Examples</title>

<p>Line 1 is correct because it uses a correctly-formatted value.</p>

<p>Line 2 is incorrect because it specifies an invalid value.</p>

<table id="TABLE\_825A459AB2374B5E9FB550F6BC53A03A">

<tgroup cols="3"> <colspec colnum="1" colname="col1" colwidth="\*"/> <colspec colnum="2" colname="col2" colwidth="\*"/> <colspec colnum="3" colname="col3" colwidth="\*"/>

<thead>

<row> <entry> Line </entry> <entry> AmazonOrderItemCode </entry> <entry> MerchantOrderItemCode</entry>

</row></thead>

<tbody>

<row> <entry> 1 </entry> <entry> 12345678901234 </entry> <entry> 12345678901234 </entry>

</row>

<row> <entry> 2 </entry> <entry> 9876543210 </entry> <entry> 43210987654321</entry>

</row>

</tbody>

</tgroup>

</table>

</section>

</refbody>

</reference>

## Sample Task (unconditionalized)

<task id="GUID-B0E919E6-0FBE-48BC-A920-CEDBE4D835DE" xml:lang="en-US">

<title id="GUID-2753A76A-B035-45B6-8615-81655E334EC2" outputclass="notitle">List a product</title>

<prolog id="GUID-3A49A19C-D0C6-4D00-BFAE-1ADE64256A13">

<metadata>

<othermeta name="parentId" content="200220550"/>

<othermeta name="topicId" content="201973930"/>

<othermeta name="language" content="en\_US"/>

<othermeta name="country" content="US"/>

<othermeta name="sortIndex" content="2"/>

<othermeta name="group" content="2011"/></metadata>

</prolog>

<taskbody id="GUID-F3DAB8C8-4E43-4B62-87DD-A3F5CF7E1418">

<context id="GUID-3D1CF722-1AAF-4745-8C91-E0FF7F0FA410">

<p>Follow these steps to list a product for sale on Amazon:</p>

</context>

<steps id="GUID-2A2A0D9E-1705-4D4E-8AC2-C1AE4F94646F">

<step id="STEP\_6D08D12E84C34B2A9408785E4F18B3D1">

<cmd id="GUID-4BDE1E0C-6EAD-4D19-AC1B-418BCA79796C">From the <uicontrol id="GUID-02989710-BC76-454B-B910-EBE0BC365B16">Inventory</uicontrol> drop-down menu select <xref href="/hz/productsearch" scope="external" format="html" outputclass="same\_window">Add a Product</xref> and search for the product you want to sell on the Amazon.</cmd>

<stepresult>Results are more accurate if you search for a product identifier such as a UPC, EAN, JAN, or an ISBN.</stepresult>

</step>

<step id="STEP\_04B026EFAB9341728A1F97D1198892E9">

<cmd id="GUID-0C949CCF-7A9B-4532-9436-CBC6E223AC0C">If you find the product you want to list, follow these steps:</cmd>

<substeps id="SUBSTEPS\_6995B2975AEF42ACB8C8E0051C62B4AC">

<substep id="SUBSTEP\_4C716FF4C8104A2FA518E06FFCECEBFD">

<cmd id="GUID-3673D43D-62A3-4C9E-B0F9-F412ADB923EA">Click the <uicontrol id="GUID-18BD9968-1D25-4AEF-A7E8-B318A1E8A6BC">Sell yours</uicontrol> button.</cmd>

</substep>

<substep id="SUBSTEP\_492CF705585643DBA39905D056E66583">

<cmd id="GUID-B824FE36-AEE0-47FD-A8B7-31564252AA73">Add your offer details including the following information: </cmd>

<info>

<ul id="UL\_A0F31A31AE5942408FE285C3F3EA854C">

<li id="LI\_2C18BEEA3A934329971181D16AFE4CD3"> Price</li>

<li id="LI\_42105951D955472CA916B4216B1D6054"> Quantity</li>

<li id="LI\_EE1EC1FB4CDD4C6EA511B8F0C1511878"> Condition</li>

<li id="LI\_769E42DE6CCA4861B4928FFE0BE7C788"> shipping options</li>

</ul>

</info>

</substep>

<substep id="SUBSTEP\_323AD2A642DF4B3DA3B8F6AF414B43D7">

<cmd id="GUID-0CC1A6F5-8484-4685-8387-B241C976863C">Save your changes.</cmd>

</substep>

</substeps>

</step>

<step id="STEP\_DE5A72EA618A44DF93483A8924FA3425">

<cmd id="GUID-02BDACDB-EC1C-43EA-A4C6-F86E12B0FC8F">If you are not able to locate your product you can submit product information so Amazon can create a new product detail page. Detail pages are shared with other sellers who may offer the same product. </cmd>

<substeps id="SUBSTEPS\_B039A117748348429D1DDA7F2BDFAEBA">

<substep id="SUBSTEP\_5A145E1C7E3A4D588E30FAC2AF9168F6"><cmd id="GUID-696DFEA4-4D7A-4117-BCAB-05756B9ED621">Search or browse for the category that matches the product you want to offer and click the <uicontrol id="GUID-E291F647-B78D-4AD1-BFAE-7928BEFB6298">Select</uicontrol> button. Being more precise you are with your category can help buyers find your products more easily. </cmd>

</substep>

<substep id="SUBSTEP\_60D69F8C5D184319A006673C78DD6660"><cmd id="GUID-0E20BB39-75A4-4BAA-B9FD-D084BE6E4C39">Enter the vital product information and your offer details and then save and finish your offer.</cmd>

</substep>

</substeps>

</step>

</steps>

<result id="GUID-3EDACB4B-535F-4429-81DB-718AA16EBEA3">

<p>The information you submit is published to Amazon within minutes and your offer becomes visible to customers through search and browse.</p>

<p>Newly created products typically appear on Amazon website within 15 minutes, although some details and images might not appear for 24 hours.</p>

</result>

</taskbody>

</task>