JUST. SYSTEMS

OEM Untangled: XML-Based Strategies in Creating Customized Documents for OEM Products

AN XMetaL WHITE PAPER

XMetaL®

The Challenges of OEM Documentation

For some of today's most innovative technologies, the high cost of research and development can be offset only if that technology finds its way into a diverse range of other companies' products.

In our global economy, Original Equipment Manufacturer (OEM) relationships are an important model for doing business. An OEM model enables manufacturers and their partners to quickly deliver innovative products to as many customers as possible, while customers enjoy the benefits of having a well-priced product backed by a familiar brand. For complex products, OEM relationships enable vendors to offer well-integrated solutions that incorporate components from multiple companies, but are serviced by a single vendor. For some of today's most innovative technologies, the high cost of research and development can be offset only if that technology finds its way into a diverse range of other companies' products.

Unfortunately, when it comes to documenting the product for customers (marketing materials), end-users (guides and instructions), or staff (service and reference materials), the efficiencies of an OEM model do not always carry over. The need to produce and maintain customized documentation for partners' requirements can be expensive and can slow an OEM company's expansion into new markets. Even small changes in a product can be difficult to manage in documentation, which often needs to be customized for each partner, customer, or product line to reflect variations in the following:

- Product name
- Product appearance or context in which the product is used, which could affect some or all pictures of the product
- Custom features added to the product for a particular vendor
- Core features removed from the product to simplify it for a particular partner
- Partner name, logo, and contact information
- Partner's house styling requirements, e.g. use of fonts, colors, and page partner
- Partner's house writing style and terminology
- Information on support or services available for the product
- Warnings or notices required for regulatory compliance in the partner's industry and/or geographic location

These types of changes are usually easy to make, but they are challenging to make in a way that is fast, cost-effective, and maintainable. Several business factors make the task of managing OEM documentation more challenging than it initially appears:

- There is often strong pressure for customized documentation to be ready to ship as soon as a customized product is ready, even though the core (uncustomized) documentation might be barely ready by that time.
- Both core and customized documentation may need to be translated, which can be very

expensive and lead to unnecessary delays in releasing a product.

- Products usually change many times over their lifespan, so modifications may need to be repeated for each release.

In recent years, publishing systems based on XML technologies have gained popularity. XML-based publishing systems tend to deliberately decouple content and formatting.

Fortunately, strategies and technologies are available today that put the task of efficiently customizing OEM documentation within reach. The first half of this paper describes and compares, at a high level, various methods for customizing OEM documentation. The second half of this paper details how newer customization strategies can be implemented through the DITA (Darwin Information Typing Architecture) XML standard, an increasingly popular standard for companies with complex documentation requirements.

Options for Customizing Document Formatting

A very common requirement for products sold under a partner's brand is for the documentation to use the physical styling conventions of that brand, e.g. to conform to its conventional use of fonts, text colors, header and footer styles, and page layout. Historically, this has required a writer or a layout specialist to spend considerable time applying formatting throughout a document using a desktop publishing tool. Conventional desktop publishing or word processing tools are thus regarded as having "content" and "formatting" tightly bound together.

In recent years, publishing systems based on XML technologies have gained popularity. XML-based publishing systems tend to deliberately decouple content and formatting. For example, a writer could write an XML file containing installation instructions for a product, in which there is a table. That file typically does not say what the thickness and weight of the table border should be. Instead, the convention for table borders is stored in a separate file called a **stylesheet**.

After writing the instructions, the writer puts both the XML file and the stylesheet file through a type of application called an XML **publishing system**. The publishing system generates a third file called a **deliverable**, which is the formatted version of the file that customers will see. In the deliverable, the table displays with the style of borders described in the stylesheet. In an XML-based publishing system, you can have a different stylesheet for each partner and rebrand an entire document simply by running the content for that document through a publishing system with the appropriate stylesheet.

Options for Customizing Document Content

In addition to following varying requirements for formatting documents, the documentation for different versions of an OEM product often needs to vary in content. For example, consider an OEM manufacturer that produces a cell phone, which is rebranded and sold through two different mobile carriers. The two versions of the phone are nearly identical except for the following differences:

- Product names: One is called AlphaPhone and one is called BetaPhone.
- Product appearance: The phones have different home screen backgrounds and use different icons for the same applications on the home screen.
- Product functionality: BetaPhone includes a "Language" setting to use either an English or French keyboard layout. AlphaPhone does not have this option.

A feature in XML-based content reuse and some proprietary tools called conditional text (also known as “profiling”) lets authors create a master document that contains a superset of all possible variations of the document.

- Regional requirements: AlphaPhone is sold in the United States. and its documentation must include regulatory notices from the Federal Communications Commission. BetaPhone is sold in Canada and its documentation must include regulatory notices from Industry Canada.

One strategy for customizing content is to make a copy of the document, e.g. by doing a “Save As”, and then making changes to the copy. Although this is probably the most common strategy, it results in duplication of content, more files to manage, and more fixes in the event of a change to product information.

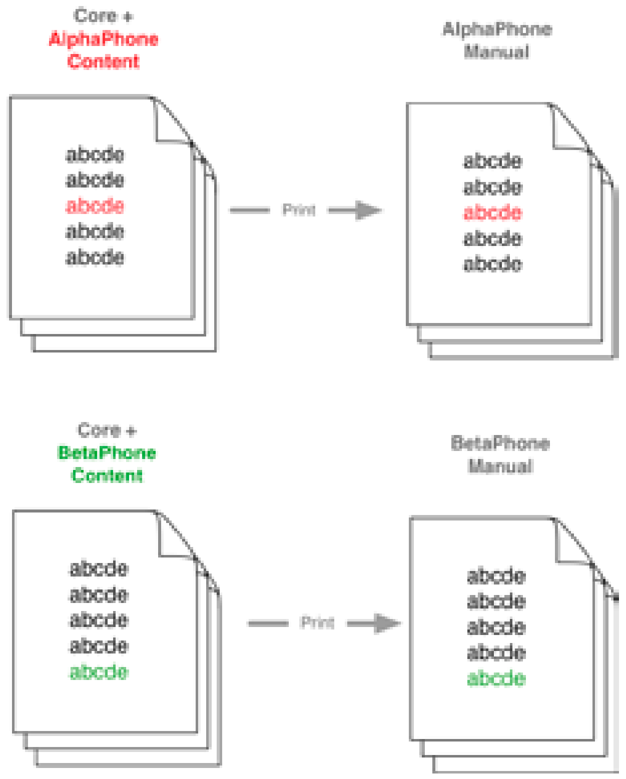


Figure 1. In the “Save As” model, core content is duplicated for each product version.

In recent decades, advances in authoring and publishing tools have enabled content to be meaningfully reused, rather than copied, across various documents. A feature in XML-based content reuse and some proprietary tools called **conditional text** (also known as “profiling”) lets authors create a master document that contains a superset of all possible variations of the document. The author can then produce a customized document by filtering out all the irrelevant variations.

With the cell phone scenario above, the master document would include:

- Text that is common to AlphaPhone and BetaPhone
- Content tagged to be included when generating the document for AlphaPhone:
  - Wherever a product name is to be shown, the phrase “AlphaPhone”
  - In the section introducing the home screen, a reference to an image file of the

Choosing whole topics to include or exclude rarely fulfills all needs for variations in OEM product documentation, but it is a useful complement to conditional text as a way to extend a document for a particular partner.

AlphaPhone home screen

- Towards the end of the document, a section on FCC regulatory compliance

- Content tagged to be included when generating the document for BetaPhone:
  - Wherever a product name is to be shown, the phrase "BetaPhone"
  - In the section introducing the home screen, a reference to an image file of the BetaPhone home screen
  - In the section on Settings, a description of the "Language" setting
  - Towards the end of the document, a section on Industry Canada regulatory compliance

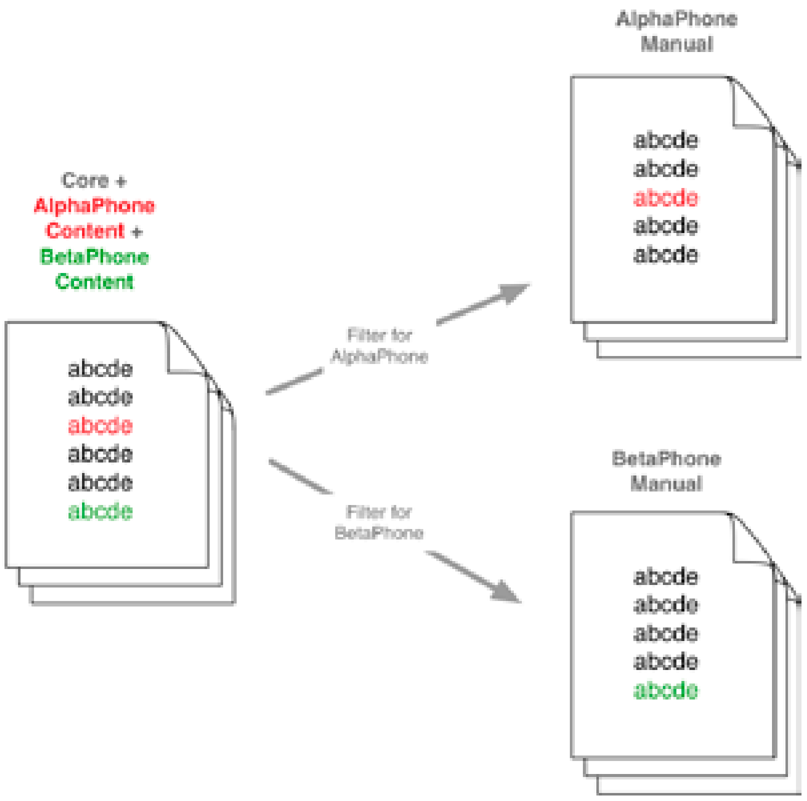


Figure 2. In the "conditional text" model, one set of source files contains a superset of all variations.

If some variations of a document need whole sections that other variations do not, authors can use XML technologies to assemble documents out of **topics**. When producing a document, an author can choose which topics to include. Choosing whole topics to include or exclude rarely fulfills all needs for variations in OEM product documentation, but it is a useful complement to conditional text as a way to extend a document for a particular partner.

Conditional text is a powerful and useful way to customize documentation, however as described later in this paper, it has limited scalability and places some limitations on the division of work within or between teams.

To create a complete, customized document, an author runs the core product package and one modification package together through an XML publishing system.

A New Paradigm: Externalized Content Customization

A relatively new architecture for delivering customized content involves segregating source files into packages, with one package of XML files for the core product and one package of XML files for each variation of the product. This strategy, which has recently become available through the DITA XML standard, tends to scale better than others for large numbers of product variations. As discussed later in this paper, it supports workflows that may be more suitable for OEM environments.

In this architecture, the package for a variation contains a set of instructions for modifying the package for the core product. To create a complete, customized document, an author runs the core product package and one modification package together through an XML publishing system. The publishing system injects changes that are specified in the “modification package”, into the “core package”. The concept is somewhat similar to the “variables” feature that is available in some proprietary authoring tools, however it allows for a greater variety of modifications and enables the modifications to be stored as a set.

With the cell phone example above, the core package would include:

- Text that is common to most variations of the phone
- Image files that are common to AlphaPhone and BetaPhone
- A section on FCC regulatory compliance

The modification package for AlphaPhone would include:

- Instructions to use “AlphaPhone” as the product name
- An image file for the AlphaPhone home screen
- Instructions to use an image file of the AlphaPhone home screen as the home screen image

The modification package for BetaPhone would include:

- Instructions to use “BetaPhone” as the product name
- An image file for the BetaPhone home screen
- Instructions to use an image file of the BetaPhone home screen as the home screen image
- A description of the “Language” setting
- A section on Industry Canada regulatory compliance, with instructions to use it in place of the section on FCC regulatory compliance

As a general rule, the more tightly content and formatting are bound together, the more effort will be needed for reformatting.

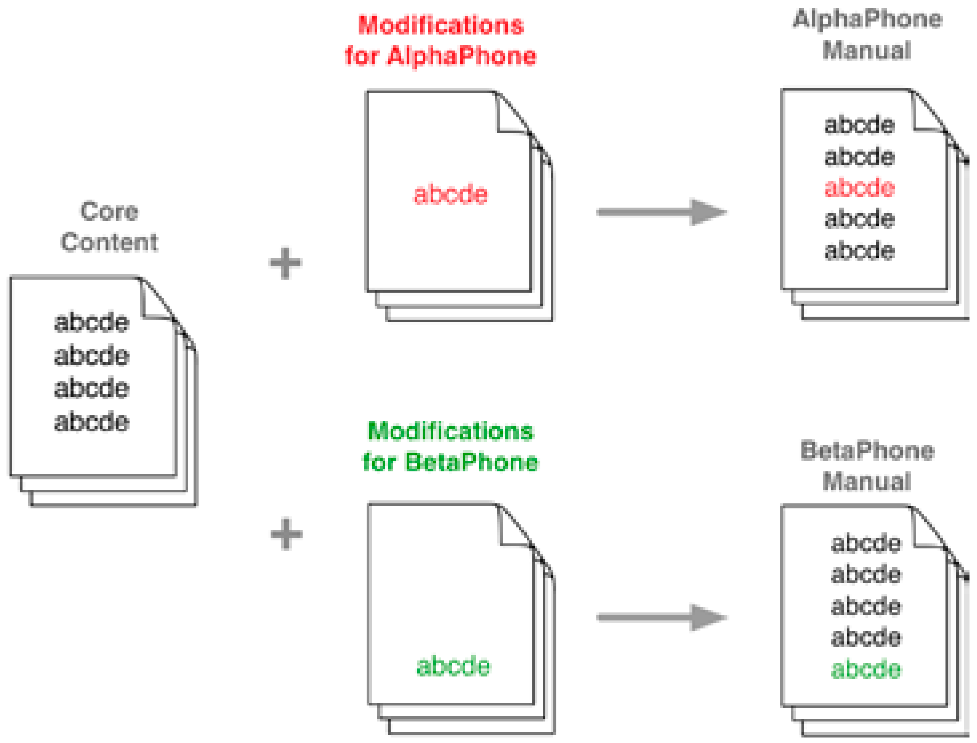

Modifications for AlphaPhone
AlphaPhone Manual
abcde
abcde abcde abcde abcde abcde
Core Content
abcde abcde abcde abcde
+
Modifications for BetaPhone
BetaPhone Manual
+
abcde
abcde abcde abcde abcde abcde

Figure 3. In the "externalized reuse" model, there is one copy of core content and it is combined with a package of modifications.

How Customization Strategy Affects Your Business

Choosing the appropriate strategy for customizing documents has considerable impact on the cost of producing and maintaining documentation, especially in the long term. In OEM business models, it can also have substantial effects on product release dates (i.e. time-to-market), documentation quality, and project manageability.

Formatting

Technology choices can make it either easy or very difficult to reformat documentation for OEM partners. As a general rule, the more tightly content and formatting are bound together, the more effort will be needed for reformatting. The most difficult approach is to use a tool in which writers lay out each page by hand and compose text to fit within boxes in the page layout.

XML-based publishing systems are the gold standard for separation of content and formatting. After an initial investment in developing stylesheets, formatting a document can be mostly or fully automated, meaning that it can be done in minutes rather than days. The practice of "separation of content and formatting" is also conducive to publishing in multiple output formats. For example, you can use XML stylesheets to quickly repurpose a document for delivery as web pages rather than as a printed guide.

Core Product Updates

Customizing a document after doing a "Save As" is easy, but if the document describes a product, a

XML-based content reuse methods avoid the problem of having multiple copies of documentation to update. When a core product is updated, the core content can be updated in just one place, and customized versions can then be re-generated easily.

problem very quickly arises: Products change. Manufacturers constantly come up with improved versions of older products and the documentation should be updated to keep pace. If the documentation has been branched into multiple varying copies, the effort needed to update all the copies can be overwhelming. The “Save As” strategy therefore has a low initial cost, but often results in high delayed costs for maintenance.

“If multiple copies of a document need to be maintained, it is also more difficult to make sustained improvements in the quality of content. Editors and writers might often spend effort in one project to improve the way a concept is explained or the way a procedure is worded and then discouragingly encounter the old text again in the next project. Often, writers and editors do not find all the explanations that need to be updated.

Starting before 1980, we kept a spreadsheet to account for all the versions of a topic we tracked in multiple documents. The most junior staff writer was responsible for updating the topic everywhere it appeared. Frequently, something was missed even though the spreadsheet provided a checklist. When I first read about a component content management system, I thought I had gone to heaven because we could update once and reuse everywhere. The OASIS DITA standard has increased the opportunities.”

Dr. JoAnn Hackos, Director of the Center for Information-Development Management

XML-based content reuse methods avoid the problem of having multiple copies of documentation to update. When a core product is updated, the core content can be updated in just one place, and customized versions can then be re-generated easily.

Translation

If customization is based on a “Save As” model, translation costs are typically far higher than with XML-based content reuse, for two reasons: First, content that is duplicated between files must either be re-translated or at least assessed for translation. Second, the separation of content and formatting in XML-based documents greatly simplifies the formatting of translated content into finished deliverables. For further information, see the JustSystems whitepaper, *Understanding and Communicating the Financial Impact of XML & DITA* .

“A key benefit with DITA adoption is to increase your organization’s flexibility to meet rapidly changing business requirements. This flexibility provides a positive impact for customer experience improvements and bottom line revenues.”

Chip Gettinger, VP, XML Solutions, SDL

New Product Variations

As a healthy OEM business grows, it tends to distribute each of its products in a growing range of variations. For example, if a company produces a refrigerator that is distributed under five different labels, it may at some point distribute the refrigerator under a sixth label as well.

When the time comes to create the sixth version of documentation for the refrigerator, a “Save As”

One of the key advantages of an externalized reuse model is that it scales extremely well.

strategy will get the job done fairly quickly, albeit with the delayed maintenance costs described above. If the company uses conditional text for product variations, it is almost always possible to add a condition type for a new product variation.

There are some significant drawbacks with using conditional text for product variations, however. First, as the number of product variations increases, writers must work with increasingly complex-looking documents.

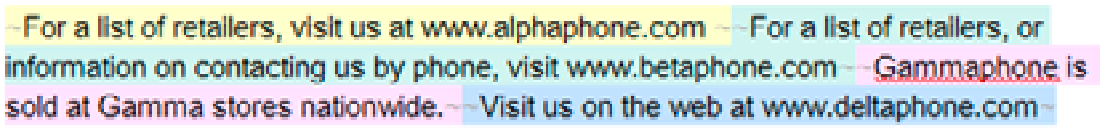

For a list of retailers, visit us at www.alphaphone.com For a list of retailers, or information on contacting us by phone, visit www.betaphone.com Gammaphone is sold at Gamma stores nationwide. Visit us on the web at www.deltaphone.com

Figure 4. If there are many variations of a product, it can become difficult to keep track of conditional text.

If mistakes are made in assigning condition tags to content, or in configuring the publishing system to filter conditional content, it can be difficult to identify and debug the issue.

One of the key advantages of an externalized reuse model is that it scales extremely well. For the sixth refrigerator distributor, it is easy to create a sixth modification package. The existing modification packages serve as patterns that writers can easily adapt to create a new one. Adding a seventh, or a thirty-seventh, product variation is just as straightforward.

Workflow and Distribution of Work

A ubiquitous issue in OEM businesses is that the people who are most familiar with the core product are seldom the people who are most familiar with how the product is customized for a particular brand and market. In fact, these individuals often work for different companies.

There are two typical ways to address this issue, neither of which is maintainable. One approach is for the OEM manufacturer to do a “Save As”, send a copy of the documentation to the partner, and have the partner make modifications. Again this is a seemingly-quick solution, but with enormous delayed costs. Furthermore, it can lead to delays in releasing the modified product, as the core product document needs to be as stable as possible before the “Save As” step is done.

Another approach is for the partner to send instructions to the manufacturer about what modifications are needed, and for the manufacturer to be responsible for producing a customized document using conditional text. This is often the best approach for small and easily-communicated changes, such as changing a product name.

However, if extensive ad-hoc changes must be made and reviewed using conditional text, significant overhead becomes needed to co-ordinate the effort between two companies. If multiple partners request customization of content that is in the same file, it is difficult for even a tight team of writers to divide the work of implementing changes to that file.

Externalized reuse can be used regardless of whether the partner or the OEM manufacturer produces the customized document:

Reduced overhead and faster time-to-market are not the only benefits of having writers work in parallel on their respective areas of expertise. A less-obvious set of benefits comes from better alignment between the distribution of work and the business model

- If the partner produces the customized document, writers from both the OEM manufacturer and the partner can work in parallel, with each focused on their area of expertise. The OEM manufacturer focuses on documenting the core product. When the OEM manufacturer has an early draft of the core documentation ready, the partner can start to create a modification package.
- If the OEM manufacturer produces the customized document, the OEM manufacturer can have a member of its writing team be responsible for understanding all the changes that are necessary for a particular partner, creating a modification package for that partner, and having the partner review the changes. In the meantime, other writers can document the core product. This division of labor minimizes bottlenecking.
- When both the core documentation and the modification package are ready, either the OEM manufacturer or the partner can merge them into one set.

Reduced overhead and faster time-to-market are not the only benefits of having writers work in parallel on their respective areas of expertise. A less-obvious set of benefits comes from better alignment between the distribution of work and the business model:

- Having a single point of contact for each partner tends to lead to better communication and more satisfactory partner relationships.
- All of the changes that have been made for a particular partner are in one place–the partner's modification package–for optimal visibility of what changes have been made and what changes the partner needs to review.
- Managers can easily track the time spent on each modification package and assess how this cost affects the profitability of each partner relationship.
- As the OEM business acquires more partners, it is straightforward to expand the writing team's capacity for handling the increased number of variations.

Risk Management

If authors use a “Save As” strategy to create a document for one partner based on a document that has already been customized for a different partner, a significant risk is that the author could forget to remove or replace content in the second partner's document, when it was intended to appear only in a the first partner's document. If the partners are competitors, this kind of error can be extremely embarrassing. The risk of this type of error also exists with a conditional text strategy.

Another type of risk arises in “Save As” scenarios, when the core product changes and multiple copies of the document need to be updated to reflect those changes. These changes are extremely tedious to implement in multiple copies and may be done incorrectly or incompletely. When the OEM partner is responsible for producing customized documentation, it is common to fall behind in the task of updating it to reflect changes in the core product.

Summary

In an OEM business model, reusing rather than duplicating content is an economical and manageable approach to customizing documentation for multiple variations of a product while minimizing risk. Cost savings increase with the longevity of the product, the requirements for translation, the length of the document set, and the number of product variants.

Like most XML-based documentation standards, the DITA standard requires content to be separated from formatting, which enables a document to be easily republished in different formats and with different styles.

"Our vision with DITA is to get to a point where we have true single-sourcing, with all information in one place."

Pamela Lu, Lead Technical Writer, EFI

XML-based publishing enables a high degree of automation for reformatting documents and offers many options for reusing content. A customization strategy based on conditional text is suitable when there are just a few product variations and documentation is maintained by a single team, however it is cumbersome to scale for larger numbers of product variations in OEM environments. A customization strategy based on externalized reuse scales more readily and is easier to manage in a large team, while significantly reducing the risk of errors in the content.

Implementing Content Customization in DITA XML

Various technologies for reusing content are available based on the DITA XML standard. Conformance to the DITA standard requires that products from different vendors fulfill certain requirements for interoperability, meaning that the various writers, illustrators, and translators working on a document do not all have to use tools from the same supplier.

Like most XML-based documentation standards, the DITA standard requires content to be separated from formatting, which enables a document to be easily republished in different formats and with different styles. It also supports conditional text and topic-based document assembly. For further information on conditional text in DITA, see the presentation Customizing Content with DITA Conditional Text and XMetaL®.

Version 1.2 of the DITA standard introduced support for externalized reuse through features called *keyref* (including a subfeature of *keyref* called *conkeyref*) and *conref push*.

The following sections describe how the DITA conkeyref and conref push features can be used to customize core content by injecting changes from a modification package. The code samples shown here are only for illustrating concepts and are not necessarily fully detailed. The same results could also be achieved in DITA through slightly different markup, but for the sake of simplicity only one approach is shown here.

"We work with many information development teams that are responsible for creating multiple versions of their content for subsidiaries or partner organizations. In the past, the work has been a burden because the teams are rarely resourced for the additional work required. With DITA 1.2 and indirect referencing, we advise clients to create master topics and use keyrefs, conkeyrefs, conref push, and other reuse strategies to accommodate a multitude of variable content."

Dr. JoAnn Hackos, Director of the Center for Information-Development Management

Background: The Structure of DITA Documents

To see how customization in DITA works, it is necessary to first know a few concepts in how DITA documents are structured.

In the DITA standard, you almost always create a document as several small files and then assemble

A common way to set up source files in DITA would be to have one DITA topic file for each chapter, and one DITA map file to indicate how to order the topics. It is recommended to have each topic in its own file.

them into one seamless document by putting the files through a DITA publishing system. The practice of writing a document as several small source files rather than as one big one makes it easier to divide work between team members, and gives teams more flexibility to reuse and customize content.

Topics and Maps

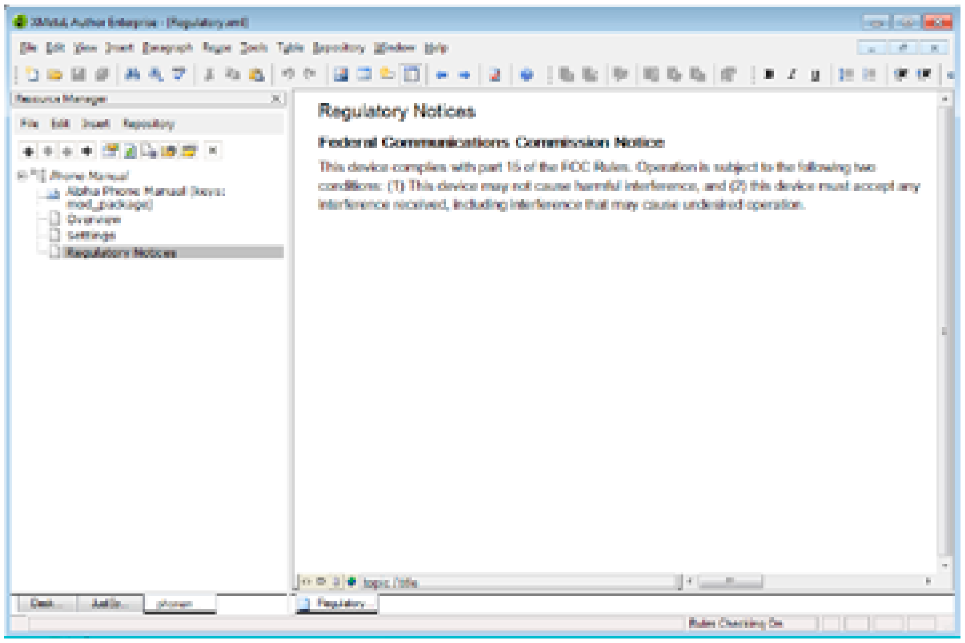
For the cell phone example introduced above, let's assume for simplicity that the manual has three short chapters, called "Introduction", "Settings", and "Regulatory Notices". A common way to set up source files in DITA would be to have one DITA topic file for each chapter, and one DITA map file to indicate how to order the topics. It is recommended to have each topic in its own file. The core document package would therefore include four files:

- *Overview.dita*
- *Settings.dita*
- *Regulatory.dita*
- *PhoneManual.ditamap*

The *PhoneManual.ditamap* file contains a reference to each topic:

```
<topicref format="dita" href="Overview.dita"></topicref>
<topicref format="dita" href="Settings.dita"></topicref>
<topicref format="dita" href="Regulatory.dita"></topicref>
```

Instead of requiring authors to look at and edit XML code directly, modern DITA authoring tools such as XMetaL® Author Enterprise provide a graphical user interface for editing topics and maps in a WYSIWYG-type environment:


XMetaL Author Enterprise - [Regulatory.xml]
File Edit View Insert Paragraph Pages Tools Table Repository Window Help
Resource Manager
File Edit Insert Repository
Phone Manual
Alpha Phone Manual [keys: mod_package]
Overview
Settings
Regulatory Notices
topic /title
Regulatory

Regulatory Notices

Federal Communications Commission Notice

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Figure 5. XMetaL® Author Enterprise provides a graphical user interface, similar to a word processor, for editing XML documents.

In the DITA standard, authors can assign a unique identifier to nearly any element.

Element IDs

XML files, like HTML files, contain multiple elements, which can vary in size from a single word to a page or more. In the following example, a <section> element contains a <title> element and a <p> element. The name “p” is an abbreviation of “paragraph”.

```
<section id="sound">
<title>Sound</title>
<p>You can set up sounds to use as a ringtone and as an alert when you receive text
messages.</p></section>
```

In the DITA standard, authors can assign a unique identifier to nearly any element. In the above example, the ID of the <section> element is “sound”. Element IDs are not visible to the customer who buys the cell phone and reads the manual, but they act as crucial markers for adding or replacing content at specific locations in the document.

Images

As with HTML documents, images in DITA documents are always stored as separate files. DITA topics include references to image files. For example, the following statement will result in the *start.png* image appearing in the middle of a sentence.

```
<p>Press <image href="start.png"></image> to begin.</p>
```

Customization Example 1: Changing Product Names and Images

Some of the changes between variations of an OEM product are predictable. In the cell phone example introduced above, the team creating the core documentation set can probably predict that each variation of the phone will have a different name (e.g. AlphaPhone and BetaPhone) and a different home screen.

For these predictable changes, a good strategy is described below. You create one package for the core content for the phone, a modification package for AlphaPhone, and a modification package for BetaPhone. Each modification package contains a DITA map file, one or more topic files that are referenced from the DITA map, and image files that are referenced from the topics. It is convenient to place the files from a particular package into a folder, although this is not necessary.

The name of the conkeyref attribute is an abbreviation of “content reference via a key”.

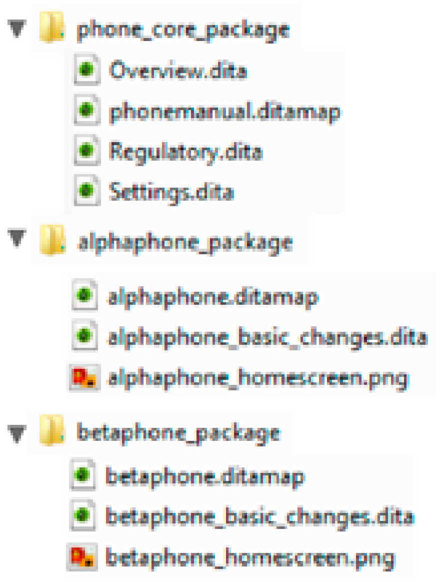


Figure 6. Files in packages for changing product names and images

Topics and Image Files in the Core Document Package

The three topics in the core content package include all text and references to all images that are common to both variations of the phone.

The topics contain a placeholder each time the product name appears. A sentence containing a placeholder for the product name looks like this:

```
To turn on your <ph conkeyref="basic_changes/productname">cell phone [Placeholder
for cell phone product]</ph>, press the rectangular button at the top of the unit.
<p>Press <image href="start.png"><alt>Start</alt></image>to begin.</p>
```

The name of the *conkeyref* attribute is an abbreviation of “content reference via a key”. This placeholder indicates that text should be brought in from an element whose ID is *productname* in a topic called *basic_changes*. The text “cell phone” will appear in the unlikely event that the placeholder cannot be resolved.[1]

The Overview topic, which describes the home screen, includes a placeholder for the picture of the home screen. The sentence introducing the home screen looks like this:

```
The home screen appears: <image conkeyref="basic_changes/homescreen"></image>
```

The placeholder above indicates that the path to the image file should be brought in from an element whose ID is *homescreen* in a topic called *basic_changes*.

Topics and Image Files in the Modification Packages

The modification package for AlphaPhone contains a small topic file to define the two placeholders described above. The name of the file is *alphaphone_basic_changes.dita* and it includes the following content:

[1] It is also possible to change product names using the <i>keyref</i> feature of DITA rather than the <i>conkeyref</i> feature. However, the <i>conkeyref</i> feature is more flexible in that it works for both images and text.</fn>

```
<ph id="productname">AlphaPhone</ph>
<image id="homescreen" href="alphaphone_homescreen.png"></image>
```

Each modification package includes one DITA map file.

The modification package for AlphaPhone also contains an *alphaphone_homescreen.png* file.

The modification package for BetaPhone contains a topic file named *betaphone_basic_changes.dita* with the following content:

```
<ph id="productname">BetaPhone</ph>
<image id="homescreen" href="betaphone_homescreen.png"></image>
```

The modification package for BetaPhone also contains an *betaphone_homescreen.png* file.

If you recall, the *conkeyref* attributes in the topics in the core package reference a topic called *basic_changes*, not the file names *alphaphone_basic_changes.dita* or *betaphone_basic_changes.dita*. The next section describes how the name *basic_changes* is assigned to the topics in the modification packages.

Maps in the Modification Packages

Each modification package includes one DITA map file. The AlphaPhone modification package includes a file called *alphaphone.ditamap*, with the following contents:

```
<title id="manual_title">AlphaPhone Manual</title>
<topicref keys="basic_changes" href="alphaphone_basic_changes.dita" />
```

The attributes on the topicref element indicate that the *basic_changes* topic is found in the file named *alphaphone_basic_changes.dita*.

The BetaPhone modification package includes a file called *betaphone.ditamap*, with the following contents:

```
<title id="manual_title">BetaPhone Manual</title>

<topicref keys="basic_changes" href="betaphone_basic_changes.dita"/>
```

Map in the Core Document Package

The core content package contains one DITA map file, which includes:

- References to all the core topics that will go into a deliverable.
- A reference to the map in the appropriate modification package.
- A placeholder title for the manual, which will be replaced by a title from the modification package.

References to the core topics look like this:

```
<topicref format="dita" href="Overview.dita"></topicref>
<topicref format="dita" href="Settings.dita"></topicref>
<topicref format="dita" href="Regulatory.dita"></topicref>
```

The reference to the appropriate modification package looks like this if you are generating the manual

The conkeyref feature of DITA is extremely flexible, allowing you to replace virtually any element or topic with a modified version.

for AlphaPhone:

```
<mapref format="ditamap" keys="mod_package" href="../alphaphone_package/alphaphone.
ditamap"processing-role="resource-only"/>
```

Note that there is an attribute saying *processing-role="resource-only"*. This indicates that the topics in *alphaphone.ditamap* should not be directly included in output. The contents of these topics are only to be used as a resource in other topics.

If you are generating the manual for BetaPhone, the reference to the appropriate modification package looks like this:

```
<mapref format="ditamap" keys="mod_package" href="../betaphone_package/betaphone.
ditamap"processing-role="resource-only"/>The placeholder title for the manual looks
like this, and will pull in the appropriate title for the overall publication from
the map in the modification package: <title conkeyref="mod_package/manual_title">Phone
Manual</title>
```

Assembling Content from Core and Modification Packages

When the map file in the core product package is put through a DITA publishing system, the core product content will be automatically combined with the content from the modification package for either AlphaPhone or BetaPhone, depending on which one is referenced in the core map file.

We have heard from many customers that key-based content reuse is very important for their i nformation architecture plans. With XMetaL® Author Enterprise 7.0, our support for keyref/conkeyref simplifies the creation and testing of relationships between pieces of content.

Addam Smith, Director of Research and Development, JustSystems Canada, Inc.

Other Types of Content Replacements

The *conkeyref* feature of DITA is extremely flexible, allowing you to replace virtually any element or topic with a modified version. For further information, see the article *DITA 1.2 Keyref: Feature Description*. *If content needs to be translated, it is generally recommended that if you use placeholders within sen-tences, that they be used only for proper nouns, as sentence structure varies between languages. Some organizations set complete sentences as the minimum conditionalization boundary. Full guidelines for making DITA content suitable for translation are given in the articles on Optimizing DITA for Translations by the OASIS DITA Translation Subcommittee.*

Customization Example 2: Making Ad-Hoc Additions or Changes

In the cell phone scenario, there are two types of ad-hoc changes:

- In the topic on Regulatory Notices, AlphaPhone has a section on FCC regulatory compliance, whereas BetaPhone has a section on Industry Canada compliance.
- In the "Settings" topic, BetaPhone has a "Language" setting that other phone variations do not have.

You can store ad-hoc changes in your modification packages and have them injected into the appropriate places in the core content. For example, you can create a topic file containing the ad-hoc changes for a particular product variation, store it in the folder for that product variation, and reference it from the map file for that variation:

If most variations of a product require a certain piece of content in their documentation, it might be simplest to have that content be in the core content package and replace it only when needed.

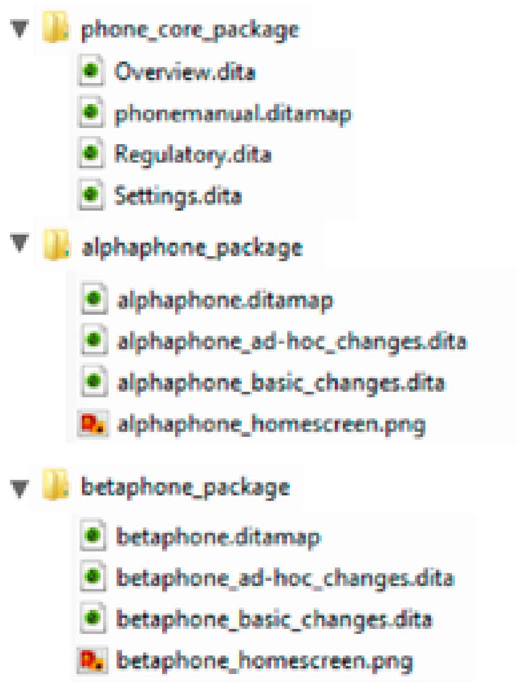


Figure 7. You can add a topic file to each modification package to store ad-hoc changes.

Replacing a Section Using a Modification package

If most variations of a product require a certain piece of content in their documentation, it might be simplest to have that content be in the core content package and replace it only when needed. For example, say that most variations of a cell phone require paragraphs on United States FCC regulatory compliance, and BetaPhone is one of the few variations sold in other countries. For this case, it may be most convenient to include statements on FCC regulatory compliance in the core content and have the BetaPhone modification package say to replace the section about FCC regulatory compliance with a section about Industry Canada Compliance.

The *Regulatory.dita* file contains a section about FCC regulatory compliance. To make it possible to replace this section, it must have an ID attribute:

```
<topic id="regulatory">
…
<section id="FCC_notice">
<title>Federal Communications Commission Notice</title>
<p>This device complies with part 15 of the FCC Rules. Operation is subject to the
following two conditions: (1) This device may not cause harmful interference, and (2)
this device must accept any interference received, including interference that may
cause undesired operation.</p></section>
…
</topic>
```

To facilitate ad-hoc changes, the core package owner can ensure that elements in the core package have element IDs.

In the BetaPhone modification package, a file called *betaphone_ad-hoc_changes.dita*, contains the following:

```
<section conref="../phone_core_package/Regulatory.dita#regulatory/FCC_notice"
conaction="pushreplace">
<title>Industry Canada Notice</title>
<p>Operation is subject to the following two conditions: (1) this device may not cause
interference, and (2) this device must accept any interference, including interference
that may cause undesired operation of the device.</p></section>
```

The *conref* attribute contains a path to a topic and the ID of the element in the core package that contains the section on FCC regulatory compliance. The *conaction* attribute says what to do with that section.

Finally, in the *betaphone.ditamap* file, the following statement ensures that the topic containing ad-hoc changes is included as a resource when generating output:

```
<topicref href="betaphone_ad-hoc_changes.dita"/>
```

To facilitate ad-hoc changes, the core package owner can ensure that elements in the core package have element IDs. XMetaL Author Enterprise includes an option to automatically assign unique IDs to all elements of particular types. For example, if you want to let any <section> element be replaceable via a modification package, you can set up a preference to have XMetaL Author Enterprise automatically assign an ID to every <section> element. Alternatively, the modification package owner can ask the core package owner to add IDs to the core package where needed.

When updating the core document, e.g. for a subsequent version of the product, ensure that elements used as hooks for ad-hoc changes do not have their IDs changed or deleted. Also ensure that those element IDs are not moved or copied to locations where the ad-hoc changes don't make sense. It will be helpful if the writers who change the topics in the core package are aware of what parts of the core content have been changed through ad-hoc, externalized changes. You can keep track of these locations by adding comments, using the <draft-comment> element. To quickly check that ad-hoc changes still make sense after a document is updated, you can generate a PDF file of the customized deliverable and search for the text of the changes that you have pushed into it.

Adding a Section from a Modification Package

If a product variation has a unique feature, you can push documentation about that feature from a modification package into the appropriate place in the document. Say that BetaPhone has a unique "Language" setting and it should be described in a section immediately after the section called "Sound". To make it possible to inject a section immediately after the "Sound" section, this section in the *Settings.dita* file must have an ID attribute:

```
<topic id="settings">
….
<section id="sound">
<title>Sound</title>
<p>You can set up sounds to use as a ringtone and as an alert when you receive text
messages.</p></section>
…
</topic>
```

In the BetaPhone modification package, you would add the following to the *betaphone_ad-hoc_changes.dita* file:

If multiple product variations require the same change, you can reference a single file containing that change into multiple maps.

```
<section conref="../phone_core_package/Settings.dita#settings/sound"
conaction=”mark”></section>
<section conaction="pushafter">
<title>Language</title>
<p>Select the city closest to your current location, and your preferred language for
keyboard input.</p></section>
```

The first <section> element above is empty and its sole purpose is to indicate where the second <section> element should go. In the second <section>, the attribute *conaction =”pushafter”* indicates that the section should be added after the marked section in the core content.

Other Types of Ad-Hoc Changes

The method described above for adding or replacing content in the core document set is known as *conref push*, which is an abbreviation of “pushed content reference”. In addition to using this method to add or replace sections, you can use it to add or replace nearly any type of element, such as paragraphs, numbered list items, table rows, or whole topics. As with *conkeyref*, if content needs to be translated, do not modify phrases within sentences, except for proper nouns.

“XMetaL Author Enterprise 7.0 integrates DITA 1.2’s expanded reuse potential for our customers with an easy-to-use interface. Our customers will find increased customer satisfaction, increased productivity, and improved quality and accuracy of information documents across the organization.”

Tim Groeneveld, Vice President of Sales and Marketing, JustSystems Canada, Inc.

If multiple product variations require the same change, you can reference a single file containing that change into multiple maps. For example, if AlphaPhone and BetaPhone both have a particular feature but GammaPhone does not, you can create a DITA topic file containing the changes related to that feature and reference that file into both the AlphaPhone and BetaPhone maps.

There are some rules and restrictions that the DITA standard places on what you can add to a document using this feature, the most significant of which is that you can use it to add a maximum of two elements at any particular location. If you need to add more two elements, e.g. if you want to add 5 rows to the bottom of a table, you should replace the entire table instead of adding rows to it.

Another limitation is that if you use it to add a new element of a particular type, the core document must already contain an element of the same type either immediately before or immediately after the location where you want to add the element. For example, you can insert a paragraph after a paragraph or a table after another table, but you cannot insert a table between two paragraphs.[2] For further information on the *conref push* feature, see the article *DITA 1.2 Feature Overview: Conref Push*.

Adopting XML in OEM Businesses

A wide variety of organizations have adopted DITA in recent years and yet more have adopted other XML languages. Although the reuse strategies described in this paper are powerful enough for large organizations with complex requirements, DITA-based solutions can be adapted for organizations of any size and budget. For OEM manufacturers that would like to have partners take on more responsibility for customizing documentation, externalized reuse in DITA allows companies to exchange and share source files even if many aspects of their DITA publishing systems are different.

“I teach that having a carefully designed reuse strategy is a key to a successful DITA implementation. A strategy ensures that you will meet your Return on Investment predictions and even achieve more than you thought possible. DITA 1.2 indirect referencing mechanisms like keyref and conkeyref increase your

ability to manage change."

Dr. JoAnn Hackos, Director of the Center for Information-Development Management

One of the first steps in moving to an XML-based customization strategy is to understand the patterns of what pieces of content are currently being substantially duplicated and could be reused instead. After assessing requirements for all content, (including both technical documentation and marketing documentation if there is overlap between them), an organization can formulate a strategy for chunking content into topics and files, and a strategy for when to use each reuse technique. A content chunking strategy and careful selection of reuse techniques, e.g. when to use *conkeyref* placeholders, when to do ad-hoc modifications via *conref push* and when to use conditional text, will make a big difference in how efficiently an organization can work. Many companies need to reuse content between similar core products, as well as between variations of a product for different partners.

This first stage, known as content modeling or content analysis, is often best undertaken with mentoring from an experienced outside expert. The next stage is usually a small pilot project to test how to apply reuse features to your own content and at the same time test tools for authoring, storing, and publishing content. Tool vendors can often help to set up and troubleshoot a pilot project. For more information on content analysis and pilot projects, see the articles *Planning for DITA Success - Part One: How to Set Up the Right Team and the Right Strategy* and *Planning for DITA Success - Part Two: How to Deploy DITA, Step-By-Step*.

After the initial learning curve for the team, a well-chosen reuse strategy should be straightforward for even new writers to use. If predictable variations such as product name changes are all that is required, a writer could create a modification package for a third variation (e.g. GammaPhone) by copying from either the AlpaPhone or BetaPhone modification packages. A good reuse model makes it easier for teams to grow and to collaborate with other teams, making OEM businesses well-positioned to deliver quality products through a growing network of partners.

Acknowledgments

JustSystems is grateful to the following individuals for their assistance with this article: JoAnn Hackos of Comtech Services, Chip Gettinger of SDL, James Hom and Megan Bock of NetApp, Pamela Lu of EFI, Stephon Johns of JANA, Inc., Bernadette Willis and Stella Hackell of Juniper Networks and Joe Pairman of HTC. *making DITA content suitable for translation are given in the articles on Optimizing DITA for Translations by the OASIS DITA Translation Subcommittee.*

XMetaL®
JUSTSYSTEMS.

http://www.justsystems.com

North America:
Phone: +1.866.793.1542
Email: ContactSales-NA[at]justsystems.com

Europe:
Phone: +44 (0) 1462 889 082
Email: ContactSales-EMEA[at]justsystems.com

International: +1.604.697.8705

About JustSystems

JustSystems is a leading global software provider with over three decades of successful innovation in office productivity, information management, and consumer and enterprise software. With over 2,500 customers worldwide, the company is continuing a global expansion strategy based on its XMetaL content lifecycle solutions, the world's most complete and user-friendly XML authoring tool. JustSystems has been received many industry accolades including: KMWorld's 100 Companies that Matter in Knowledge Management in 2012, EContent 100 Companies that Matter the Most in Digital Content in 2011, KMWorld's Trend Setting Product of 2011, Software Magazine's The Software 500 of 2008, and many others. Major strategic partnerships include IBM, Siemens, and EMC. For more information, visit: www.justsystems.com.