How does OASIS DITA support content reuse?

Stanley Doherty, Ph.D.

Guidepost

Purpose: This resource provides a conceptual overview of the ways that OASIS DITA 1.3 supports content reuse by reference.

Learning objective(s):

- Identify what sorts of content can be reused by reference in DITA.
- Recognize what to do to make a piece of content reusable.
- Review how to reference that reusable content into DITA topics.
- Illustrate the difference between direct (URL-based) and indirect (key-based) referencing mechanisms.
- Review some industry best practices for scalable reuse.

Prerequisites:

- Familiarity with basic concepts associated with structured content.
- Familiarity with W3C XML elements and attributes.
- Familiarity with DITA maps, topics, semantic markup, and linking.
- Familiarity with creating DITA objects in a DITA editor.

Revision history (ACM-013)

Date	**Author**	**Revision summary**
TBD	Stan Doherty	First public posting

What is content reuse?

In the **informal sense**, writers reuse content when they copy content from one topic, revise it, and then paste it into a different topic. This informal reuse results in multiple, slightly variant instances of the original content. These many instances can become a maintenance burden over time. We call this technical debt, that is content that we know that we'll need to go back later to fix. In the **formal sense**, content reuse requires that there be only one instance of a piece of content anywhere in a source filesystem and that this one instance gets referenced – not copied – into multiple other topics.This is often called single-sourcing content. Managing this reusable content may increase team overhead, but it also increases the return on investment (ROI) in working with structured content.

Introduction

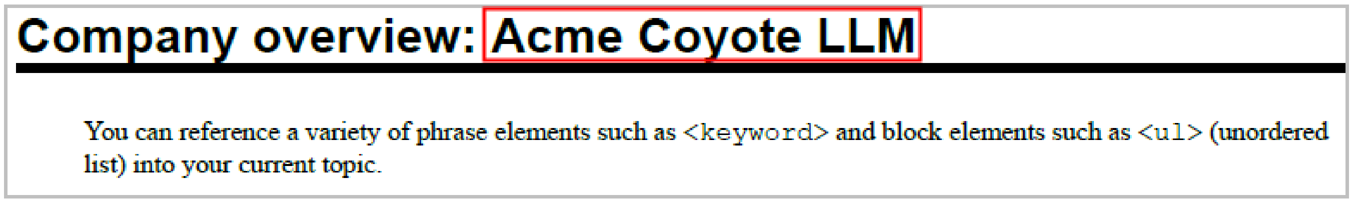
When teams invest in structured content, they are in a good position to reuse any of that content across their publications. DITA, OASIS DocBook, and formal docs-as-code environments may use different mechanisms to reuse a piece of content in multiple locations, but the underlying principles are the same:

- Assign a unique tag or ID to the piece of content in a library topic.
- Reference that tag or ID in the library topic from the current topic.

Quick example: Let's say that a writer creates a DITA 1.3 topic named `lib_phrases_company-branding.dita`. Like a public book library, this topic contains content that many users on the team can extract. These library topics are also called warehouse topics or reuse topics. This sample library topic has the XML ID `lib_phrases_company-branding` and a keyword element `<keyword id="keyword_company-name">Acme Coyote LLM</keyword>`. This keyword element holds the value of the company name, Acme Coyote LLM. Writers will likely need to reference this company name many times from more than one topic in the collection of topics. From a current topic, `intro.dita`, a writer can insert the value of that library element using the following statement:

```
<title>Company overview:
  <keyword conref="lib_phrases_company-branding.dita \
     #lib_phrases_company-branding/keyword_company-name"/>
</title>
```

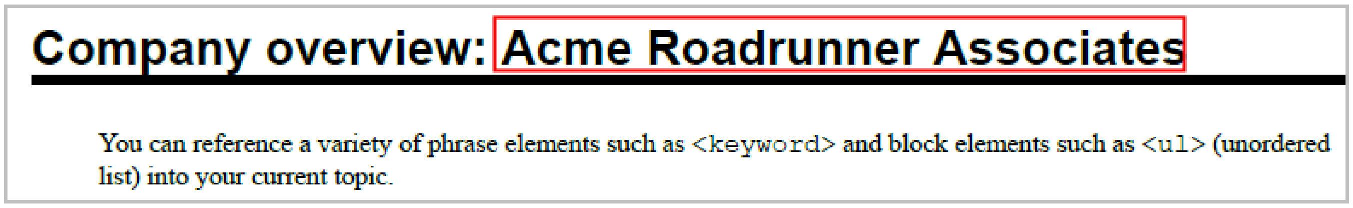
When the writer processes the topic `intro.dita`, the XML parser searches for the file named `lib_phrases_company-branding.dita`, verifies its topic ID `lib_phrases_company-branding`, and then extracts the value of the element with the ID `keyword_company-name`. In generated output, the reader sees, "Company overview: Acme Coyote LLM."



Company overview: Acme Coyote LLM

You can reference a variety of phrase elements such as `<keyword>` and block elements such as `<ul>` (unordered list) into your current topic.

If writers on the team create 100 DITA content references (also known as `@conrefs`) to this library element, a writer can update the value generated from all 100 `@conrefs` by updating the value of the reusable element in the library. For example, if the company is acquired and its new name is "Acme Roadrunner Associates," updating the name once in the source `lib_phrases_company-branding.dita` topic will update all other topics containing that keyword reference.



Company overview: Acme Roadrunner Associates

You can reference a variety of phrase elements such as `<keyword>` and block elements such as `<ul>` (unordered list) into your current topic.

Learning how to reuse content in DITA is a matter of designing libraries of reusable elements and then referencing them judiciously into topics.

NOTE: All the sample code referenced in this resource is available at: https://github.com/acm-sigdoc-structured/dita-reuse-overview/tree/main.

What sorts of content can be reused in DITA?

Basically, any element that supports the @id attribute can be reused by reference in DITA.

Element type	**Elements**	**Sample element with @id**

Maps	Generic maps
	
	Bookmaps
		`<map id="`**`map_rootmap_acme`**`"> . . .`
		
		`<bookmap id="`**`bookmap_user-guide`**`" . . .`

Topics	Generic
	
	Concepts
	
	Tasks
	
	Reference
		`<topic id="`**`topic_acme-prelims`**`"> . . .`
		
		`<concept id="`**`concept_acme-intro`**`"> . . .`
		
		`<task id="`**`task_install-acme`**`"> . . .`
		
		`<reference id="`**`ref_acme-platforms`**`"> . .`

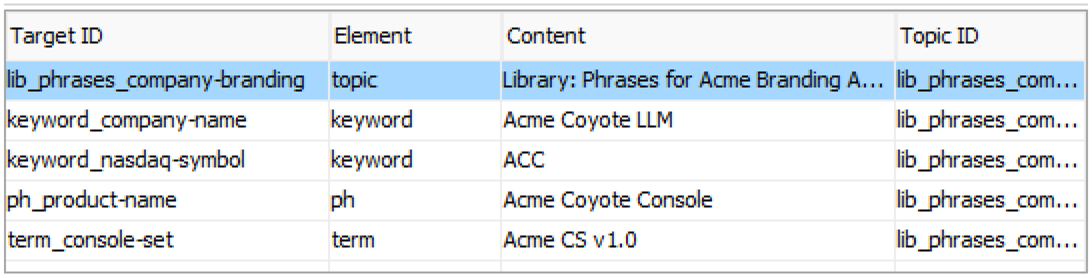
Blocks of elements	Simpletables	`<simpletable id="stable_admin-logins">`
	Steps	`<steps id="steps_login"> . . .`
	Unordered lists	`<ul id="ul_login-prereqs"> . . .`

	Figures/images	`<fig id="fig_acm-logo">`
		`  <title>ACM logo</title>`
		`  <image id="image_acm-logo"`
		`    placement="break"`
		`    class="- topic/image "`
		`    alt="Infection"`
		`    scope="external"`
		`    scale="25"`
		`    href="https://sigdoc.acm.org \`
		`      /wp-content/uploads/2024/02/ \`
		`      SIGDOC_withTag-1536x574.png"/>`
		`</fig>`

Phrase elements	Phrases	`<ph id="ph_product-name"> . . .`
	Keywords	`<keyword id="keyword_company-name"> . .`
	Terms	`<term id="term_console-set"> . . .`

If the `@id` attribute value for a piece of reusable content contains a prefix that tells the writer the name of the containing element, then the writer is able to judge whether that type of element can be referenced into the current topic at a specific location. Writers cannot insert any DITA element anywhere into a topic. For example, inserting the `<simpletable>` element into a `<title>` element invalidates the topic. Some DITA authoring tools provide a preview of all the available elements in a library, so writers can identify which of those elements that can be inserted by reference where.

In the following screen shot from a DITA editor, we see such a list of available library elements in `lib_phrases_company-branding.dita`. Note that the prefix of the element @id (displayed here as **Target ID**) identifies the type of element in the library containing the shared content, for example `term_`, `keyword_`, or `ph_`.



Target ID	Element	Content	Topic ID
lib_phrases_company-branding	topic	Library: Phrases for Acme Branding A…	lib_phrases_com…
keyword_company-name	keyword	Acme Coyote LLM	lib_phrases_com…
keyword_nasdaq-symbol	keyword	ACC	lib_phrases_com…
ph_product-name	ph	Acme Coyote Console	lib_phrases_com…
term_console-set	term	Acme CS v1.0	lib_phrases_com…

Not to worry. If a writer attempts to insert a piece of reusable content where it breaks the validity of the topic, validating DITA editors will generate an error and, in some cases, identify a location in the current topic where the writer *can* insert that element.

Phrase and block libraries

While you are thinking about how you might want to organize the content in library topics, consider segregating phrase and block elements into separate libraries.

Phrase elements typically appear in running text, do not contain subordinate elements, and do not appear in output with breaks before and after. In our previous example, we saw a library named `lib_phrases_company-branding.dita` that was full of phrase elements such as `<keyword>` and `<term>`. Developing phrase libraries is a great starting point for learning about DITA reuse. Words and short phrases that can be reused are relatively easy to identify in your content and can be inserted into most DITA contexts.

Block elements contain other subordinate elements and appear in generated output between breaks. Lists, tables, sections, paragraphs, and figures are some of the most common block elements organized into block libraries, for example `lib_blocks_company-branding.dita`. Teams that make extensive use of block libraries need to invest time in making the block content as agnostic as possible to context. References to "previous section" or "next example" would inhibit reuse in many contexts. Curating shared block content requires the attention of all team members and, in large organizations, requires formal content governance processes.

How do authors reference these pieces of reusable content?

Once a team sets up libraries containing reusable elements with IDs (typically in libraries), it can reference those reusable elements so they appear in a content topic. The technical term for this process is "transclusion" – inserting content by reference.

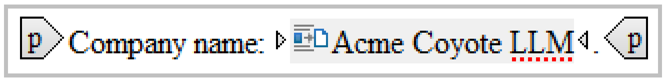
The syntax for referencing objects is as follows:

```
<in-topic-element-name
conref="filepath/library-name.dita/library-topic-id#targetelement-id">
```

To insert a content reference to the library `<keyword>` element with the `@id` `keyword_product-name`, a writer would insert the following statement inside a `<p>` element:

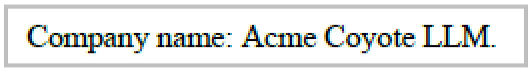
```
<p>Company name:
   <keyword conref="./lib_phrases_company-branding.dita \
   #lib_phrases_company-branding/keyword_company-name"/>
</p>
```

Here is how that appears in a DITA editor.



p Company name: Acme Coyote LLM. p

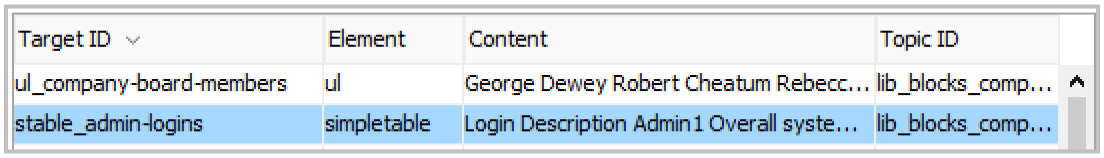
Here is how it appears in generated output.



Company name: Acme Coyote LLM.

The attribute `@conref` tells the DITA parser that it needs to insert referenced content from another DITA library. The `./lib_phrases_company-branding.dita` statement tells the parser to look in the current folder for a library topic named `lib_phrases_company-branding.dita`. The statement `#lib_phrases_company-branding/`**`keyword_company-name`** tells the parser to retrieve the value of the `keyword_company-name` element inside the library topic with the @id `lib_phrases_company-branding`. Most DITA editors will let the writer pick the library element from a list and insert the content reference element.

The following screen shot from a DITA editor illustrates how the block library `./lib_blocks_company-branding.dita` contains a ready-to-reference unordered list (`ul_company-board-members`) and simple table (`stable_admin-logins`).



Target ID	Element	Content	Topic ID
ul_company-board-members	ul	George Dewey Robert Cheatum Rebecc...	lib_blocks_comp...
stable_admin-logins	simpletable	Login Description Admin1 Overall syste...	lib_blocks_comp...

The mechanics for DITA content reuse are relatively straight-forward. The challenges tend to involve managing the shared content and making it easy for writers to understand and use.

What is indirect referencing (DITA keys) all about?

NOTE: This section discusses an advanced reuse feature in DITA 1.3. If you are just interested in learning about the basics of DITA reuse, consider skipping this section.

When we use the DITA `@conref` attribute to point to a specific library containing a reusable element, we need to include several pieces of information:

- The local filesystem path to the library, for example `./project-x`.
- The filename of the library, for example `lib_phrases_company-branding.dita`.
- The library topic @id, for example `lib_phrases_company-branding`.
- The @id of the element being referenced, for example `keyword_company-name`.

```
./project-x/lib_phrases_company-branding.dita#lib_phrases_company-branding \
/keyword_company-name
```

If the value of *any* one of these file paths, filenames, or @ids changes, every @conref to that piece of reusable content will break. Small teams of writers working on a small publication (say, under 500 topics) know where all the shared content lives and can fix the occasional breakage quickly if and when it happens. We call this form of referencing URL-based, direct, or "hard-wired" referencing because the reference contains static values that cannot change without breaking the reference. For small teams working with libraries that do not change their location or contents frequently, these URL-based references have the advantage of simplicity. Larger, distributed teams needing to change the locations or @ids of libraries incur some significant maintenance and debugging costs with URL-based @conrefs.

Enter DITA key-based referencing.

A DITA key is a named token that specifies the location of a library. When writers enter a key-based reference in a topic, they use the `@conkeyref` attribute, the name of the DITA key, and the @id of the element being referenced. We define DITA keys in separate DITA maps using the `<keydef>` element. We include a reference to those maps containing key definitions in our top-level (root) maps. This allows our XML parsers to search through all the key definitions referenced in any map and resolve the location of the desired library topic.

Quick example: The content of our sample library `lib_phrases_company-branding.dita` need not change at all. Using a direct reference (`@conref`), we can still reference a library element into our topic:

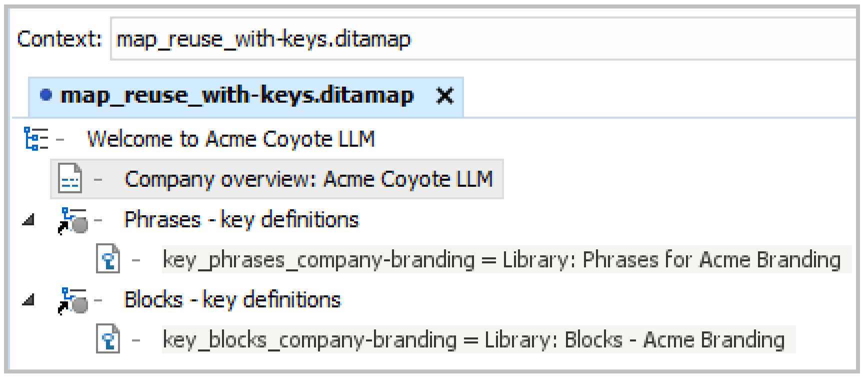
```
<p>
  <keyword conref="./lib_phrases_company-branding.dita \
    #lib_phrases_company-branding/keyword_company-name"/>
  </keyword>
</p>
```

To provide a DITA key that points to that same library, we create a separate DITA map (`keydef_phrases_company-branding.ditamap`) and reference it from our root map (`map_reuse_with-key.ditamap`). This is what a key definition looks like in one of those key definition maps.

```
<map>
    <title>Key definitions</title>
    <keydef keys="key_phrases_company-branding" \
      href="./lib_phrases_company-branding2.dita"/>
</map>
```

The key name `key_phrases_comany-branding` serves as a proxy for the actual file path and topic `@id` **`./lib_phrases_company-branding.dita#lib_phrases_company-branding`**.

Here is what a root map (**Context**) looks like when it references two key definition maps, one for a phrase and one for a block library.


Context: map_reuse_with-keys.ditamap

map_reuse_with-keys.ditamap

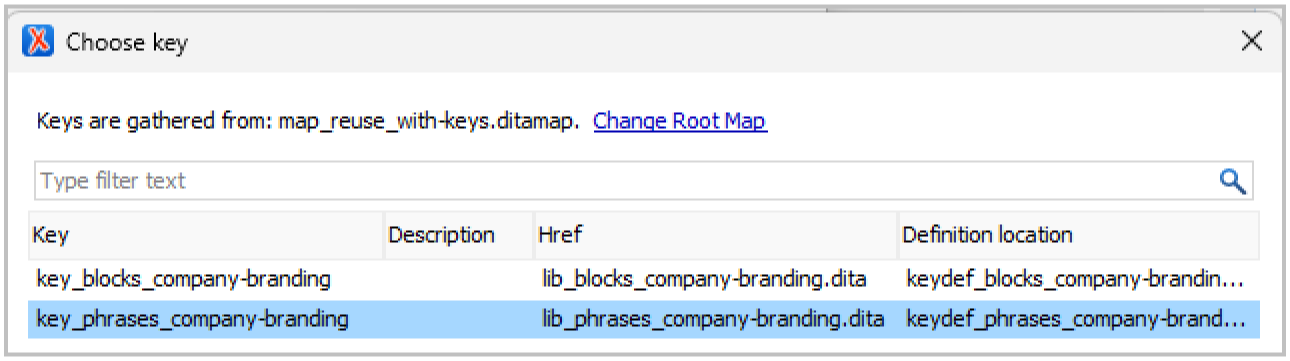
- Welcome to Acme Coyote LLM
  - Company overview: Acme Coyote LLM
  - Phrases - key definitions
    - key_phrases_company-branding = Library: Phrases for Acme Branding
  - Blocks - key definitions
    - key_blocks_company-branding = Library: Blocks - Acme Branding

These key definitions get us half way to a complete key-based reference. Whereas a URL-based reference (`@conref`) provides complete filepath and @id information to the library (**`./lib_phrases_company-branding.dita#lib_phrases_company-branding)`**, a key-based reference (`@conkeyref`) substitutes the key name (**`key_phrases_company-branding).`** Both URL-based and key-based references include the `@id` of the library element being reused (`keyword_company-name`).

URL	`<p>`
	  `<keyword conref="./lib_phrases_company-branding.dita \`
	    `#lib_phrases_company-branding/keyword_company-name">`
	  `</keyword>`
	`</p>`

Key	`<p>`
	  `<keyword conkeyref="key_phrases_company-branding/keyword_company-name">`
	  `</keyword>`
	`</p>`

When writers want to insert a key-based reference, they typically ask their DITA editor to display the key names that are referenced in key definition maps referenced from their root maps.



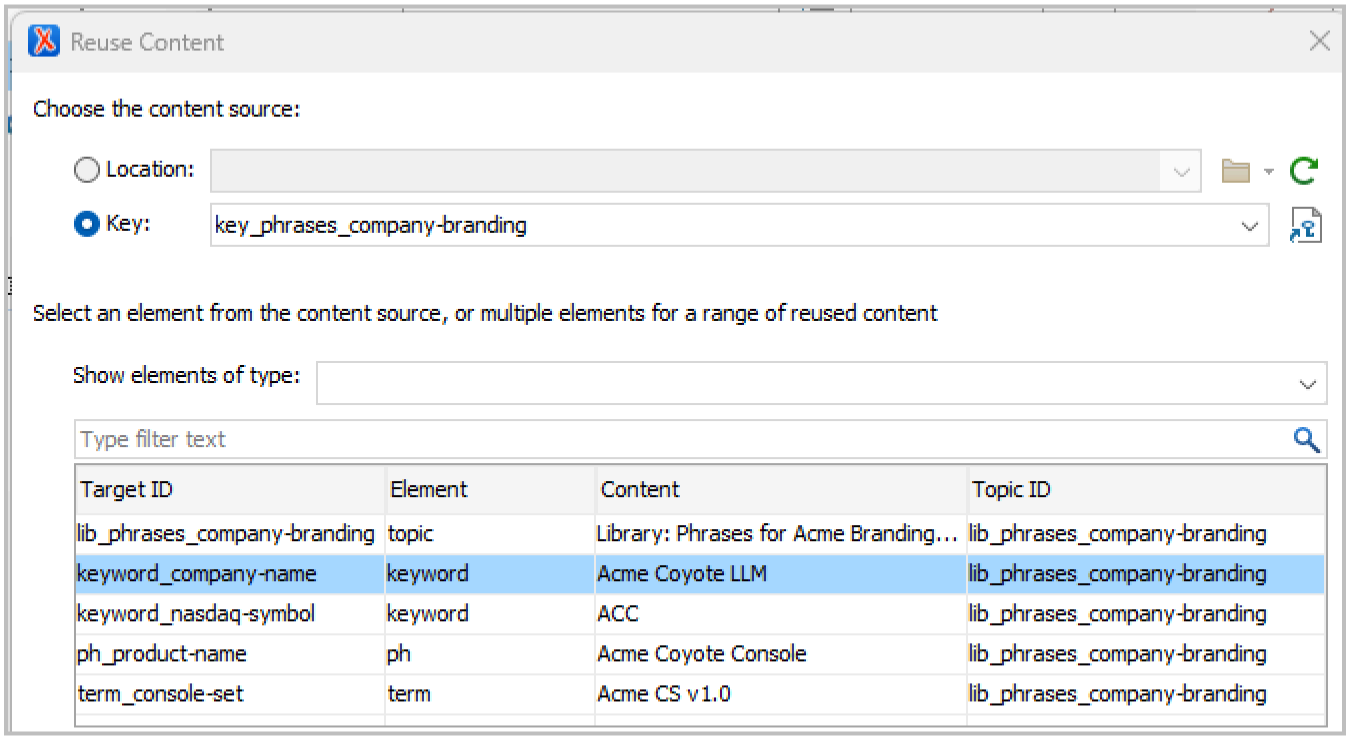
Choose key

Keys are gathered from: map_reuse_with-keys.ditamap. Change Root Map

Key	Description	Href	Definition location
key_blocks_company-branding		lib_blocks_company-branding.dita	keydef_blocks_company-brandin...
key_phrases_company-branding		lib_phrases_company-branding.dita	keydef_phrases_company-brand...

In this example, the two key definitions (`key_*`) point to two corresponding library files (`lib_*.dita`).

To insert a phrase element, the writer selects the appropriate key name (`key_phrases_company-branding`). DITA editors then typically display the reusable element @ids contained in that library file.



Reuse Content

Choose the content source:

Location:

Key: key_phrases_company-branding

Select an element from the content source, or multiple elements for a range of reused content

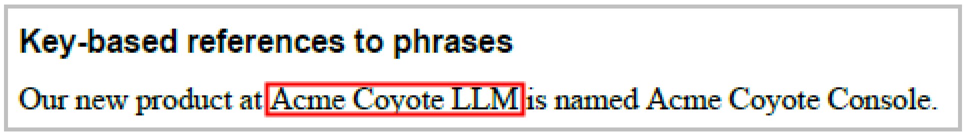
Show elements of type:

Target ID	Element	Content	Topic ID
lib_phrases_company-branding	topic	Library: Phrases for Acme Branding...	lib_phrases_company-branding
keyword_company-name	keyword	Acme Coyote LLM	lib_phrases_company-branding
keyword_nasdaq-symbol	keyword	ACC	lib_phrases_company-branding
ph_product-name	ph	Acme Coyote Console	lib_phrases_company-branding
term_console-set	term	Acme CS v1.0	lib_phrases_company-branding

When the writer selects the element @id `keyword_company-name`, the DITA editor inserts the key-based reference into the current document.

```
<keyword conkeyref="key_phrases_company-branding/keyword_company-name"/>
```

When the writer processes the topic containing this statement, the XML parser searches through all the key names that are available to its current map, finds the key definition named `key_phrases_company-branding` pointing to the library file `lib_phrases_company-branding.dita`, and extracts the value of the reusable phrase element keyword `keyword_company-name`.


Key-based references to phrases

Our new product at Acme Coyote LLM is named Acme Coyote Console.

So - what are the benefits of using key-based references?

- *Key value flexibility*: Teams can update the file location, topic `@id`, or filename of libraries with the knowledge that they can redefine the value of one key name in one key definition map to propagate those changes. All key-based references automatically inherit the new value of the key and resolve the reference to the element. As the needs of the team and the contours of the content evolve, key-based referencing provides much-needed flexibility.

- *Key definition map flexibility*: Over time, product managers expand the number of versions or feature configurations for a product. That one root map for a product eventually needs alternate values for many of its reusable phrase or block elements. Teams could develop multiple root maps to address the variants or they could simply swap variant key definition maps with the same map filename into the same root map. Managing these substitutions can be a little tricky, but letting the sophistication of DITA key-based referencing do the heavy lifting is a win.

When teams are starting out in DITA, they do not typically require the power or complexity that DITA key-based referencing offers. But as the size and complexity of their content increases, knowing that DITA keys are available reduces the long-term risk and increases productivity.

What are some best practices around content reuse?

NOTE: This section discusses some advanced applications of DITA reuse. If you are just interested in learning about the basics of DITA reuse, consider skipping this section.

Several years ago I attended a retirement party for a senior writer who had been working in DITA for ten years. One of his parting comments was, "I have no use for reuse." He argued that the overhead required to design, implement, and maintain all these reuse libraries and content references was greater than the benefit.

DITA provides the most sophisticated mechanisms for content reuse in the industry – without any guarantee that teams adopt the right ones or implement them responsibly. In the same company, one team may implement a successful reuse strategy while the team just next to it has lost control. If teams do not proactively manage the complexity, the complexity tends to manage them.

Teams with successful reuse strategies do things differently from one another, but they tend to exhibit some common denominators - some best practices.

- *Establish a governance process for the shared content:* Successful teams develop a vocabulary around reuse. They are able to review current practices, develop new ones, and - most importantly - assess what is working and what is not. The larger the writing team, the more formal the governance process needs to be.

- *Reference libraries, not other topics*: Nothing in DITA prevents a writer from entering topic-to-topic references instead of topic-to-library references. That said, in production environments, topics should *always* reference reusable phrases and blocks stored in libraries. Topic-to-topic references create a morass of interdependent references - spaghetti code. When writers discover a piece of reusable content, they should move it into a library where everyone can see it as a shared piece of content and manage it accordingly.

- *Avoid library-to-library references*: Whenever possible, do not have one reusable block of content in a library reference another reusable block in a different library. Doing so creates spaghetti-code dependencies between libraries. It's OK to reference phrase-level elements such as product names across libraries, but not blocks.

- *Develop naming conventions*: Component Content Management Systems often use programmatic, non-human-readable GUIDs (Globally Unique Identifier) as `@id`s for elements. Outside these database-driven environments, what writers name a folder, map, topic, or element `@id` makes a big difference. If writers need to stop every time they want to reference a piece of reusable content to look up its location and containing element, writers will be less likely to invest time in reuse. Consistent, easy-to-understand naming conventions increase writer productivity and improve the efficiency of any automated checking that is part of the publishing process.

- *Decide on a reuse workflow:* Some teams adopt a front-ended reuse strategy. They review all the existing content as it is converted to DITA and set up the appropriate libraries. Others defer the development of reuse libraries until they have identified reuse opportunities on the back end as they go. Although it is possible for one half of a team to front-end the libraries and the other half to back-end them, that may introduce more context switching than many writers are comfortable with. Have the discussion on your team and consider leaning into one strategy or the other.

- *Organize libraries by topic/content type*: As writers work more with DITA and become familiar with its various topic types, they find that libraries work better when they are organized by topic type as well. In some cases – task topics – *must* put reusable task elements such as `<steps>` or `<context>` in a library that is an instance of the task topic type. Placing conceptual or reference information in concept or reference library topics is not required, but helps teams organize everything in the long run.

- *Annotate each library item*: When writers create a reusable phrase or block element in a library, they have a clear understanding why they did it, where it can be reused initially, and who is the library content owner. Six months or a year later, they may have moved on to another project, and writers reviewing the reusable library element may lack context. Taking a minute or two to add simple W3C-style comments (`<!-- comment -->`) about the reusable element will increase the utility and manageability of library elements.

- *Formalize editorial standards*: It will be difficult to reuse blocks of text in multiple contexts if there is a lot of variation in the way they are written. This is also a *huge* impediment to sharing content across teams with different editorial standards. Topic titles, parallel lists, and `<step>` sentence syntax are common points of inconsistency.

- *Writing styles*: There is no universally accepted writing style in the industry. We may share many conventions, but there will always be a certain amount of variation. A block of reusable content may blend nicely into one context and not another. Looking at the writing style of reusable content with an eye to making it blend more generally is a good conversation to have.

- *Develop automation to monitor compliance*: Structured content such as DITA lends itself to automated checking. With sophisticated frameworks such as Schematron, teams can convert their reuse conventions into a set of rules that DITA editors or build systems can use to check DITA content. For example, if the team decides that all `@conref` references should point to library topics, they can write a Schematron rule that generates a message, warning, or error if someone creates a `@conref` to another content topic (instead of a library topic). Teams can set the severity of any rule violation. A message or warning notifies the writer that something needs to be fixed. This gives writers some leeway if they forgot a rule or simply ran out of editing time before a deadline.

- *Develop a sandbox domain*: It is often useful to create and maintain a set of DITA maps, topics, and libraries that reflect your team's reuse conventions. New writers can refer to it as a model. Architects can experiment with it to test new ideas – before going near production content. New team members bring great ideas to the party. Encourage engagement by asking them to demonstrate their ideas within a sandbox domain.

- *Refactor existing content to make it reusable*: When teams are new to DITA, it is common to hear, "There are no opportunities for content reuse in our existing content." Typically that is a true statement. Content that was authored apart from a robust reuse strategy and robust tooling has not been reviewed or refactored to make it reusable across multiple contexts. Experienced DITA writers and architects can provide the mentoring to change that.

- *Document procedures and standards*: Yes. All of them.

- *Connect a team's reuse strategy to its SEO strategy*: If a team references a piece of shared content 20 times, it is very possible that the content may show up 20 times in search results. SEO (Search Engine Optimization) and documentation build systems have ways to reduce or eliminate that redundancy, but teams should reach out to their SEO and build colleagues to engage them on this redundancy issue. Redundant SEO results can be addressed and should not scuttle a content reuse strategy.

Acknowledgements

- Dr. Joann Hackos - technical review
- Dr. Carlos Evia - instructional review
- TBD - editorial review

Thank You!

See https://acm-sigdoc-structured.org to learn more about committee activities, available resources, and volunteer opportunities.

Created by members of the ACM SIGDOC Committee on Structured Authoring and Content Management