The Center for Information-Development Management
710 Kipling Street • Suite 400 • Denver, CO 80215 • 303/232-7586 • www.infomanagementcenter.com

ARTICLE REPRINT

Complex Reuse Strategies with DITA 1.3 Key Scopes

Chris Nitchie, Oberon Technologies

A Tale of Two Chrises

Somewhere in New England, a person is walking around with my name. I have never met this person. To the best of my knowledge, nobody I know has ever met this person. Other than some minor details, I don't think I have much in common with this person. Everything I know about him, I learned from Google. (Yes, I sometimes Google myself. Sue me.) And because of all that, the fact that this person shares my name has had absolutely zero impact on my life. Nobody has ever gotten us confused. There has never been a context in which the answer to the question, "where's Chris Nitchie?" could be ambiguous for either one of us.

A person's name is much like a DITA key. In DITA, referencing something using the "keyref" attribute is akin to calling a thing by its name, whereas referencing something via the "href" attribute is like giving directions to where that thing currently is. Just as a thing can go by several names, a given piece of content can have any number of keys.

The inverse, however, is not true. If that other Chris Nitchie and I were somehow in the same room, and somebody asked, "Where's Chris Nitchie," we'd both raise our hands. That person would then have several options. They could just pick one of us, more or less at random, though the results of that are unpredictable. Another option might be that the identity of the questioner makes it obvious which of us he means; if it's my brother asking, he's probably not looking for the other guy. But what if it's a stranger? In that case, the person would have to give some additional information about which of us they're looking for. She'd have to specify the name *and the context* that identifies which "Chris Nitchie" she's actually trying to find.

In DITA 1.2, if you have two topics that have the same key, you don't have any options; only one of those keys can be referenced (the first one, basically). A root map defines the context for key reference resolution, and that context must be shared by all references. This makes certain things impossible.

- If you have two DITA maps with similar sets of key names, you cannot combine them into a single publication, since many of the links in one will wind up pointing to targets in the other.
- You cannot create "mail merge" publications, where the same template topic is reused multiple times, with different resolution rules for keyrefs and conkeyrefs, resulting in different effective topics.
- You cannot reuse key names in multi-product documentation sets, even when it makes sense. For example, there are cases where it would be logical for each chapter in a book to have its own "introduction" key. That's impossible in DITA 1.2.

The key scoping mechanisms in DITA 1.3 are designed to help address these cases.

Key Scopes

DITA 1.3 introduces the "keyscope" attribute, which is available on map and topicref elements. The value of the "keyscope" attribute is a name or set of names for that scope, separated by spaces. The presence of the "keyscope" attribute establishes a key scope around that element. Key references within a key scope are resolved against the definitions in the same scope. Put another way, the "keyscope" attribute puts a virtual one-way fence around that portion of the map, from which key definitions cannot escape.

For example:

```
<map>
  <title>Scoped Keys Example</title>
  <topicgroup keyscope="scope-1">
    <keydef keys="keyName" href="key1.dita" />
    <topicref href="topic1.dita" />
  </topicgroup>
  <topicgroup keyscope="scope-2">
    <keydef keys="keyName" href="key2.dita" />
    <topicref href="topic1.dita" />
  </topicgroup>
</map>
```

This map contains two references to the topic "topic1.dita." Let's assume that this topic contains key references to "keyName." In DITA 1.2, when this map is published, both output renditions will be identical, because the *keyName* is bound to the first effective definition—in this case, key1.dita. In DITA 1.3, however, those references will result in different output for each instance of topic1.dita, because they are contained by different key scopes. The binding of "keyName" to "key1.dita" is *scoped* to the topicgroup with keyscope="scope-1," and similarly, the binding of "keyName" to "key2.dita" is *scoped* to the topicgroup with keyscope="scope-2."

This is all we need to solve the problems identified earlier:

- To combine multiple standalone publications with overlapping key names into a single, omnibus publication, we can simply set the "keyscope" attribute on the references to each map, preventing their keys from interfering with each other.
- To create a "mail merge" publication, we create a single map with key scope-defining wrappers around each instance of our template topic and the appropriate key definitions.
- To create multi-product documentation, where each section contains some common key names, we put key scopes around the parts of the map that pertain to different products.

Scope Inheritance

A key scope inherits all of the key definitions from its parent scope, so you can set different key definitions to be applicable at different levels of your map structure. You might have a set of keys that apply to an entire product line with each product having a different set of keys specific to that product.

One wrinkle to be aware of is the fact that keys from a parent scope override the keys in child scopes. This allows higher-level map authors to take control over the key bindings for those broader sections of the map. However, it might seem counter-intuitive to some who might assume that the "closest" key definition is the applicable one.

Cross-Scope Linking

For key references within a scope, it is generally obvious which definition of a given key is being requested, namely, the one from the current scope. This definition is analogous to that situation where my brother walks into a room containing both me and the other guy with my name, and asks for "Chris Nitchie." We all know which one of us he means because he shares my context. But what if, for some reason, he was actually looking for the other guy? He could say something like, "Chris Nitchie from New England," and then we'd all know. Even though he shares my context, he's able to refer to the Chris Nitchie from another context.

This same ability occurs in DITA 1.3. A key reference can specify a scope name along with the key name to reference keys from another scope.

```
keyref="scopeName.keyName"
```

That's all there is to it. To specify a key from another scope, you specify the scope name, followed by a period, followed by the key name. If the key is many scopes deep, you need to specify all of the names, from the top down to the key, relative to the current scope.

Something important to notice about that scope-qualified key reference is that it is, by itself, a valid key name, because periods are valid characters in keys. This means that if you wanted to reuse a topic containing scope-qualified key references in a map that doesn't require key scopes, you could do so by providing explicit definitions for the scope-qualified key names. For example:

```
<keydef keys="scopeName.keyName" href="someTopic.
dita" />
```

Something else to keep in mind is that a key scope can have more than one name, and scope names can also contain periods. So scope-qualified key references can be made to work in maps with different scope structures.

Cross-Publication Linking

You can think of a different publication as a different scope for linking. Just as you can use scopes to cordon off and reference different sections of a map, you can also use the keyscope attribute to configure key references to the keys in an entirely different publication. This configuration is accomplished by referencing the other publication's root map via a topicref whose scope attribute is set to "peer."

```
<mapref href="otherPublication.ditamap"
        scope="peer"
        keyscope="otherPub" />
```

Your topics can now reference the keys in the other publication via a qualified key reference, "otherPub.keyName", for instance. When published, *some* processors *might* be able to turn those references into active links in *some cases*. It depends on how much information about the various published outputs for other maps can be made available to the publishing engine and whether that publishing engine is set up to use that information appropriately. The DITA 1.3 specification doesn't guarantee that all processors will be able to handle cross-publication links, but it does prescribe the appropriate markup for authoring such links.

Conclusion

Just as that other Chris and I don't bother each other because we're separated by different geographical and social contexts, key definitions within a map can be cordoned off in their own scopes. This ability opens up a host of exciting new content reuse opportunities.