An OASIS White Paper

DITA 1.2 Keyref: Feature Description

Sowmya Kannan, Sun Microsystems

21 September, 2009

OASIS (Organization for the Advancement of Structured Information Standards) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. Members themselves set the OASIS technical agenda, using a lightweight, open process expressly designed to promote industry consensus and unite disparate efforts. The consortium produces open standards for Web services, security, e-business, and standardization efforts in the public sector and for application-specific markets. OASIS was founded in 1993. More information can be found on the OASIS website at http://www.oasis-open.org.

The OASIS DITA Adoption Technical Committee members collaborate to provide expertise and resources to educate the marketplace on the value of the DITA OASIS standard. By raising awareness of the benefits offered by DITA, the DITA Adoption Technical Committee expects the demand for, and availability of, DITA conforming products and services to increase, resulting in a greater choice of tools and platforms and an expanded DITA community of users, suppliers, and consultants.

DISCLAIMER: All examples presented in this article were produced using one or more tools chosen at the author's discretion and in no way reflect endorsement of the tools by the OASIS DITA Adoption Technical Committee.

Table of Contents

Keyref Overview – DITA 1.2

DITA 1.2 introduces the "keyref" feature which provides an indirect addressing mechanism to enhance portability of content.

Purpose

DITA 1.1 has powerful features to enable content reuse, such as related links, cross references and conrefs. Extensive use of these features results in a loss of portability; meaning, it makes it difficult to move, rename, break up or combine topics. When a topic is moved, every reference to the topic will need to be updated to reflect the new location of the content.

DITA 1.2 introduces the "keyref" feature which provides an indirect addressing mechanism. You can use either topicref elements or keydef elements (new with DITA 1.2) to define keys in a DITA map. Topics can now be given a symbolic name (keys attribute) that points to a topic file path (href attribute). Future references to such topics are made using a key reference (keyref attribute). At a later point in time, if the topic is relocated, the path needs to be updated only in the map where it is defined. All other references will automatically pick up the new location.

The "keyref" feature also enables sophisticated conditional processing (profiling) of content. In DITA 1.1, conditional processing is accomplished via the use of attributes such as product, platform, and audience. Attribute-based conditional processing is a powerful feature that enables content reuse, but in a large set of a documents, content can become difficult to maintain due to the number of variables. The use of keys greatly simplifies authoring and production of conditional content. We will explore this aspect in detail in subsequent sections.

We shall explore the capabilities of the keyref feature using examples. See inline comments in code samples.

Defining Keys

You can define keys in the same DITA map you use to build the navigation (TOC) for your deliverable, or you can create and reference another map dedicated to defining keys. You can use either topicref or keydef element to define keys. Use a keydef element

- when you do not want the topic to be included in the navigation. By default, the processing-role attribute for the keydef element is set to "resource-only". This means that the topic itself will not be built; it only will be used to resolve references. If you want the topic referenced by the keydef element to be built, set the processing-role attribute to "normal".
- if you are using the keydef element to define variables.

Use a topicref element when you want the referenced topic to be included in the navigation generated by the DITA map. By default, the processing-role attribute for the topicref element is set to "normal". This means that the topic will be included in the navigation and generated as output by the processor.

Using Keyref In Related Links

Related links to topics may use indirect addressing via keyrefs. Keys defined in a ditamap may be referenced from related links using the keyref attribute. Consider the following DITA map. It uses a topicref element to define a key, since the author wants the referenced topic to be displayed in the navigation.

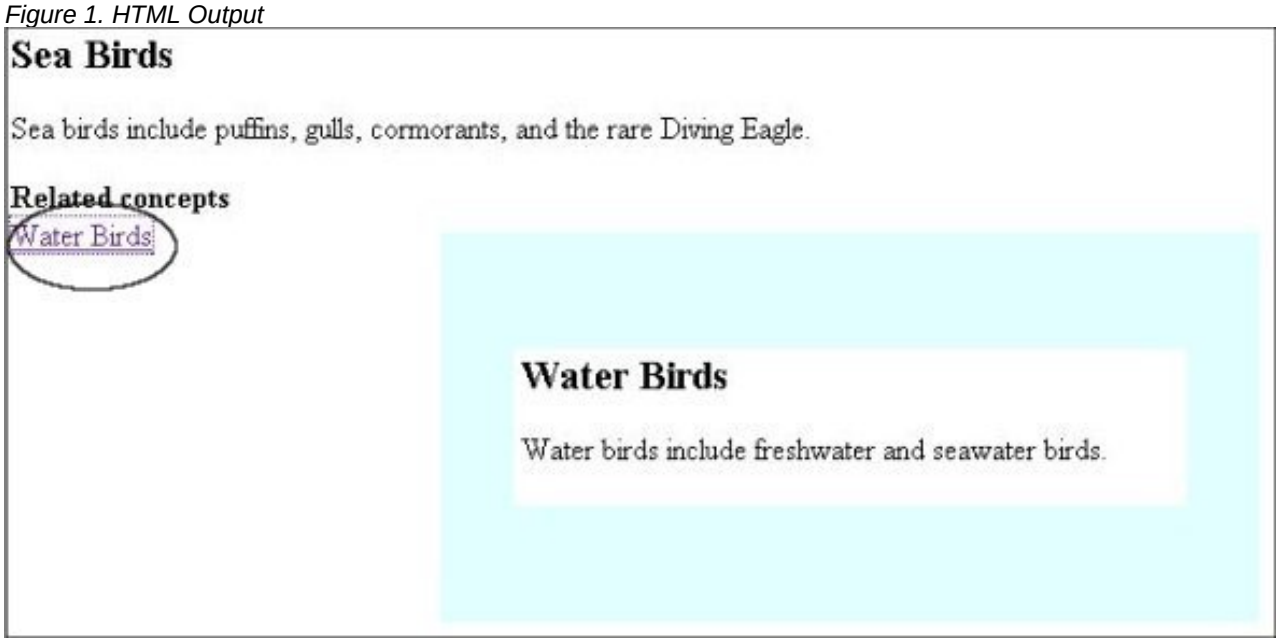
Keyref.ditamap

```
<map>
<title>Bird Guide</title>
<!-- key defined for waterbirds.dita topic -->
<topicref keys="water" href="waterbirds.dita"/>
<topicref href="seabirds.dita"/>
</map>
```

seabirds.dita

```
<concept id="seabirds" xml:lang="en-us">
<title>Sea Birds</title>
...
<conbody>
</conbody>
<related-links>
<!-- Related link uses indirect addressing with keyref to waterbirds.dita-->
<link keyref="water"/>
</related-links>
</concept>
```

Figure 1. HTML Output



Now, assume that other writers want to reuse seabirds.dita in a different map (Keyref2.ditamap). The writers want to point to their own version of the Water Birds topic; they also do not want to include the Water Birds topic in the navigation. This can be accomplished as shown below:

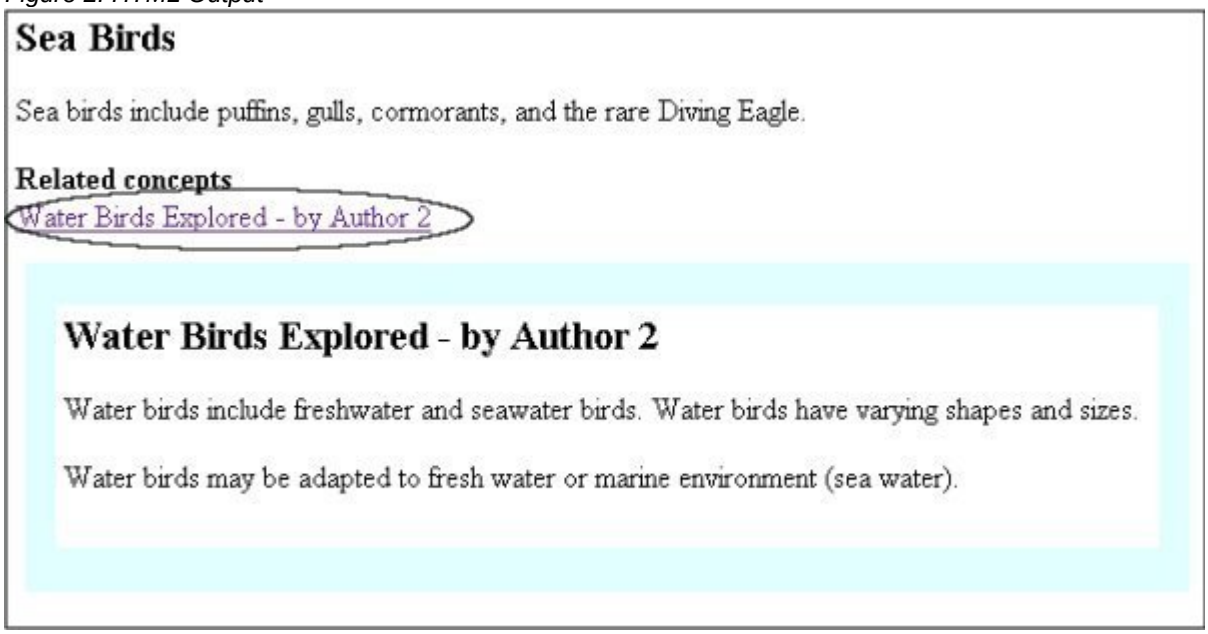
Keyref2.ditamap

```
<map>
<title>Bird Guide (2)</title>
<!-- "water" key now points to waterbirds2.dita -->
<keydef keys="water" href="waterbirds2.dita" processing-role="normal"/>
<topicref href="seabirds.dita"/>
</map>
```

Redefining a key in a different map ensures that the corresponding keyref will now resolve to a different topic.

The related link in seabirds.dita will now point to waterbirds2.dita

Figure 2. HTML Output



Complete code for this example can be found in the Keyref__UseCase1_RelLink.zip file, which is located in the sampleCode directory.

Creating Links From Terms And Keywords

An author may associate keys with glossary entries etc. in a map. These entries may be linked from terms or keywords in topics.

Keyref.ditamap

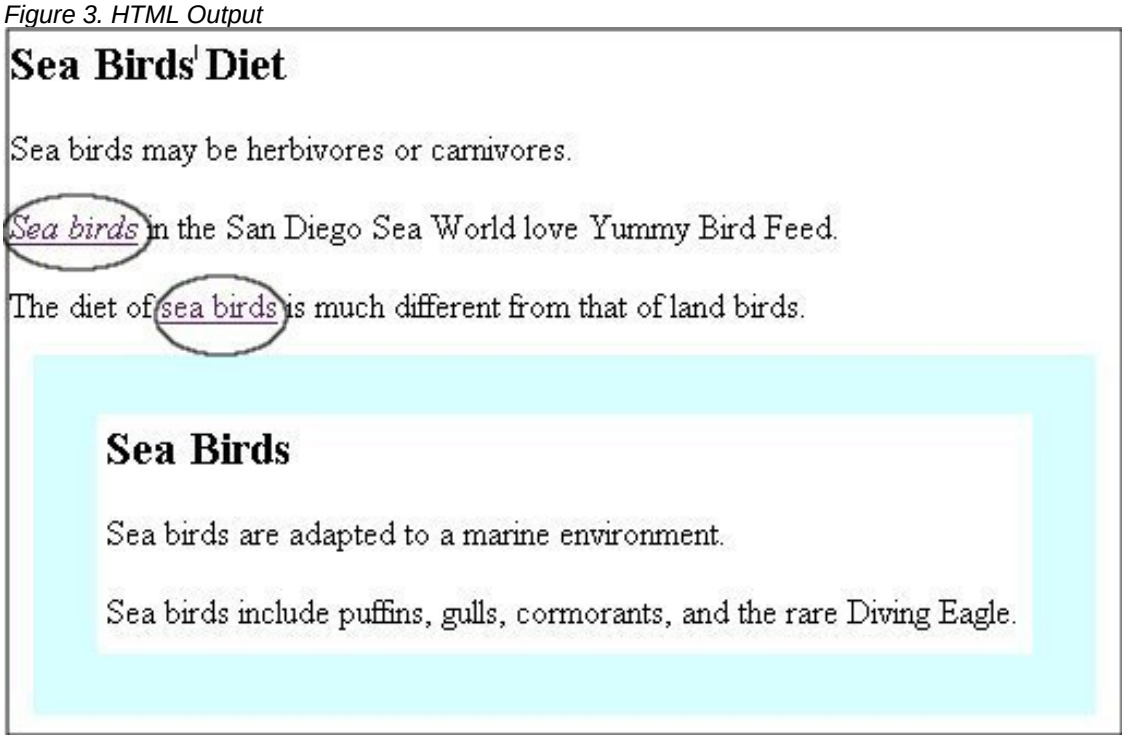
```
<map>
<title>Bird Guide</title>
<!-- Key defined for glossary term topic -->
<!-- Using topicref element because seabirds.dita is needs to be included in the navigation (TOC) -->
<topicref keys="seabirdsterm" href="seabirds.dita"/>
```

```
<topicref href="seabirdsdiet.dita"/>
</map>
```

seabirdsdiet.dita

```
<concept id="seabirdsdiet" xml:lang="en-us">
<title>Sea Birds' Diet</title>
...
<conbody>
<!-- keyref used to create links -->
<p><term keyref="seabirdsterm">Sea birds</term> in the San Diego Sea World love Yummy Bird Feed.</p>
<p>The diet of <keyword keyref="seabirdsterm">sea birds</keyword> is much different from that of land
birds.</p>
</conbody>
</concept>
```

Figure 3. HTML Output



Complete code for this example can be found in the Keyref__UseCase3_LinkFromTerm.zip file, located in the sampleCode directory.

Swapping Out Variable Content

The keydef element can be used in combination with the keyword element in a ditamap to define and set "variables" for small pieces of content that may change fairly often, such as UI labels, product name, version etc.

These keys (or variables) can be redefined later in the same map or in another map and all topics that reference the keys will automatically pick up the changes.

For example, you can put a particular product name into a DITA map, and indicate in topics where you want the product name to appear. When you generate output using the map, its topics will display the product name in the appropriate places. For longer pieces of content that do not fit into the keyword element, use the conkeyref feature instead.

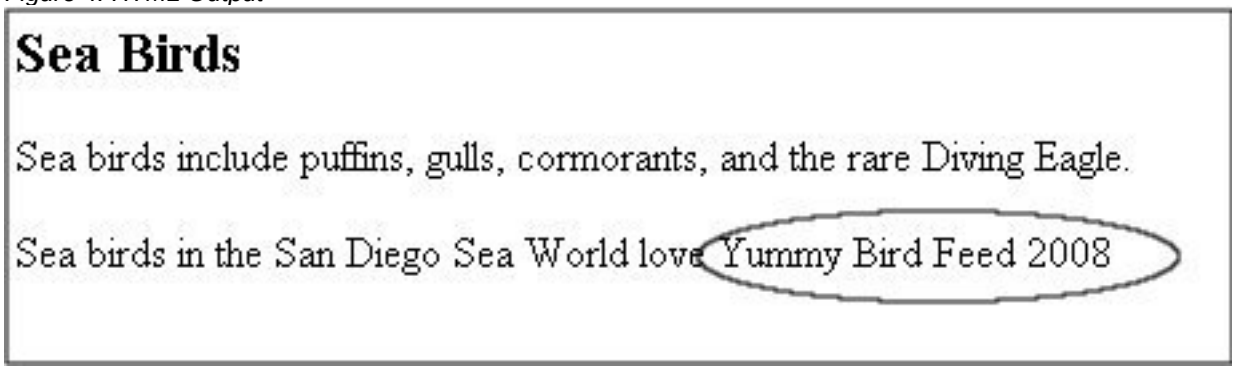
Keyref.ditamap

```
<map>
<title>Bird Guide</title>
<keydef keys="prodnameYBF">
<!-- prodnameYBF = "Yummy Bird Feed"-->
<topicmeta>
<keywords>
<keyword>Yummy Bird Feed</keyword>
</keywords>
</topicmeta>
</keydef>
<keydef keys="prodnameYBFVer">
<!-- prodnameYBFVer = "2008" -->
<topicmeta>
<keywords>
<keyword>2008</keyword>
</keywords>
</topicmeta>
</keydef>
<topicref href="seabirds.dita"/>
</map>
```

seabirds.dita

```
<concept id="seabirds" xml:lang="en-us">
<title>Sea Birds</title>
<shortdesc>Sea birds include puffins, gulls, cormorants, and the rare Diving Eagle.</shortdesc>
<conbody>
<p>Sea birds in the San Diego Sea World love <keyword keyref="prodnameYBF"/>
<keyword keyref="prodnameYBFVer"/></p>
</conbody>
</concept>
```

Figure 4. HTML Output



Complete code for this example can be found in the Keyref__UseCase4_KeywordSwap.zip file, located in the sampleCode directory.

Producing Custom Documentation For Similar Subjects

Companies often need to produce variations of a document that have the same core structure but different content for each variant. Further, these variants may be authored by multiple writers in different groups. It becomes challenging to maintain consistency between the variants. In DITA 1.1, this scenario would have been addressed using conditional processing attributes. DITA 1.2 introduces a new way to achieve conditional processing using keys - conkeyref. Let's explore the two ways of solving this use case.

Consider an example, where a company needs to produce a variation of a Bird Guide for every sea bird. Each Bird Guide will have some content common to all birds, and some content specific to a particular bird.

Using Conditional Processing Attributes To Produce Custom Documentation

In DITA 1.1, varying outputs can be generated from the same topic by specifying conditional processing attributes on elements (product, platform, and audience).

birdfood.dita contains a conditional processing attribute value for every variant:

```
<concept id="birdfood" xml:lang="en-us">
...
<conbody>
<p>Bird food is an important element in the well being of birds.</p>
<!-- Add a line for every bird; in ditaval file include desired bird; exclude others -->
<p product="albatross" conref="albatrossfood.dita#albatrossfood/fooddesc">Food for albatross.</p>
</conbody>
</concept>
```

birdhealth.dita contains a conditional processing attribute value for every variant

```
<concept id="birdfood" xml:lang="en-us">
...
<conbody>
<p>Birds of a feather flock together, hence infections spread quickly. Preventive healtch care is important.</p>
<!-- Add a line for every bird; in ditaval file include desired bird; exclude others -->
<p product="albatross" conref="albatrosshealth.dita#albatrosshealth/healthcaredesc">care for albatross.</p>
</conbody>
</concept>
```

The DITA 1.1 approach to conditional processing has the following drawbacks:

- Several topics need to be searched and modified to include information about another variant. This becomes especially complicated when dealing with several combinations of attributes and attribute values. For example, to add information about cormorants, birdfood.dita and birdhealth.dita need to be modified to include product="cormorant".
- Ditaval files can become confusing when including and excluding combinations of attribute values.
- The author is also restricted to using the conditional processing attributes (product, platform, audience, otherprops) when specifying conditions. The props attribute needs to be specialized for other types of metadata.

Using Keys - Conkeyref To Produce Custom Documentation

In DITA 1.2, the keys - conkeyref combination addresses this problem elegantly. The conkeyref attribute is similar to the conref attribute in that, it pulls in content from a referenced topic's element. However, the conkeyref attribute refers to a topic indirectly via a previously defined key and not directly via a topic file name.

The Bird Guide scenario can be authored in DITA 1.2 as shown below.

birdfood.dita uses conkeyref with a key and element id to pull in relevant information:

```
<concept id="birdfood" xml:lang="en-us">
<!-- use conkeyref instead of conref-->
<title conkeyref="birdfoodkey/birdtitle">Bird Food</title>
<shortdesc>Bird food is tailor made for every bird.</shortdesc>
<conbody>
<p>Bird food is an important element in the well being of birds.</p>
<p conkeyref="birdfoodkey/fooddesc">Description of food for birds.</p>
</conbody>
</concept>
```

birdhealth.dita uses conkeyref with a key and element id to pull in relevant information

```
<concept id="birdhealth" xml:lang="en-us">
<!-- use conkeyref instead of conref-->
<title conkeyref="birdhealthkey/birdtitle">Bird Health</title>
<conbody>
<p>Birds of a feather flock together, hence infections spread quickly. Preventive health care is important.</p>
<p conkeyref="birdhealthkey/healthcaredesc">Description of health care for birds.</p>
</conbody>
</concept>
```

albatross.ditamap defines keys that point to topics containing informations about albatross food and health:

```
<map>
<title>Albatross Guide</title>
<!-- Keys defined for bird information -->
<keydef keys="birdfoodkey" href="albatrossfood.dita" type="concept" format="dita"/>
<keydef keys="birdhealthkey" href="albatrosshealth.dita" type="concept" format="dita"/>
<!-- Topics containing conkeyrefs that refer to the keys defined above -->
<topicref href="birdfood.dita" type="concept"/>
<topicref href="birdhealth.dita" type="concept"/>
</map>
```

albatrossfood.dita contains elements with the appropriate ids referenced by conkeyref in birdfood.dita:

```
<concept id="albatrossfood" xml:lang="en-us">
<title id="birdtitle">Albatross Bird Food</title>
<conbody>
<p id="fooddesc">Albatross food is rich in vitamins A and B.</p>
</conbody>
</concept>
```

albatrosshealth.dita contains elements with the appropriate ids referenced by conkeyref in birdhealth.dita:

```
<concept id="albatrosshealth" xml:lang="en-us">
<title id="birdtitle">Albatross Bird Health</title>
<conbody>
<p id="healthcaredesc">Albatross need to be inspected every 3 months.</p>
</conbody>
</concept>
```

The DITA 1.2 approach to conditional processing has the following advantages:

- The base content topics (birdfood.dita, birdhealth.dita in this case) do not need to be modified when adding a new variant. Only the topics specific to the variant need to be added or updated (cormorantfood.dita, cormoranthealth.dita, cormorant.ditamap). New variants will work seamlessly as long as they define the same ids referenced by the conkeyref.
- An author is not restricted to specifying conditions using conditional processing attributes. Any number of variations may be created by redefining keys in different maps.
- Ditaval files are not required, hence it becomes easier to debug output.
- Some Content Management Systems have proprietary mechanisms that simulate similar behavior by switching object ids for topics. DITA 1.2 provides a non proprietary mechanism to accomplish re-direction.

Figure 5. HTML Output

Complete code for this example can be found in the Keyref__UseCase2_Conkeyref.zip file, which is located in the sampleCode directory.

Keeping References Valid When Content Is Reorganized

In an iterative development model, documentation may evolve over time. One topic may be split into multiple topics or many topics may be combined into one. One of the biggest challenges an author faces is to check that all references to a particular topic have been updated if a topic is removed or a topic is split into many smaller topics. The indirect addressing mechanism provided by the keyref feature is particularly useful in these scenarios.

Consider an example where a lead author writes introduction, example and reference information in one topic at the beginning of a project, but plans to break the information up in the next iteration. However, there are other authors who are ready to reference the lead author's content. The lead author can address the issue by defining separate keys for each section that will be broken up later. Initially, all keys will point to the same content file. At a later point in time,the lead author can change each of the keys in the map to point to different files. Other authors should use specific keys (like seabirdsintro, seabirdsexample etc.) to get relevant information.

Keyref.ditamap

```
<map>
<title>Bird Guide</title>
<!-- define separate keys for each section that you intend to reorganize, all keys point to the same dita file -->
<keydef keys="seabirdsintro seabirdsexample seabirdsref" href="seabirds.dita"/>
<topicref href="moreseabirds.dita" type="concept"/>
</map>
```

seabirds.dita

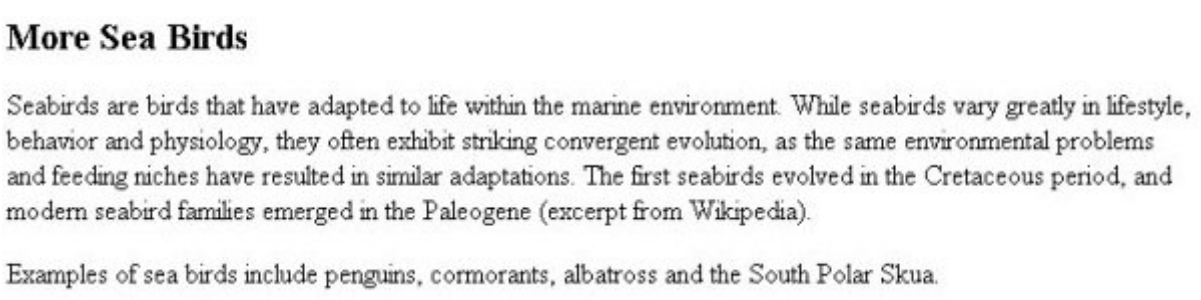
```
<concept id="seabirds" xml:lang="en-us">
<title>Sea Birds</title>
<conbody>
<!-- introduction -->
<p id="intro">Seabirds are birds that have adapted to life within the marine environment. While seabirds vary greatly in lifestyle, behavior and physiology, they often exhibit striking convergent evolution, as the same environmental problems and feeding niches have resulted in similar adaptations. The first seabirds evolved in the Cretaceous period, and modern seabird families emerged in the Paleogene (excerpt from Wikipedia).</p>
<!-- example -->
<p id="example">Examples of sea birds include penguins, cormorants, albatross and the South Polar Skua.
</p>
<!-- reference -->
<p id="reference">More information can be found on <xref href="http://en.wikipedia.org/wiki/Sea_bird"
format="html" scope="external">Wikipedia</xref></p>
</conbody>
</concept>
```

moreseabirds.dita created by another author uses relevant conkeyref (includes intro and example only):

```
<concept id="moreseabirds" xml:lang="en-us">
<title>More Sea Birds</title>
<conbody>
<!-- Other authors pull in content using relevant conkeyref, hence insulating themselves from a future
reorganization of content -->
<p conkeyref="seabirdsintro/intro"></p>
<p conkeyref="seabirdsexample/example"></p>
```

```
</conbody>
</concept>
```

Figure 6. HTML Output



Complete code for this example can be found in the Keyref__UseCase5_ReorgContent.zip file.

Abstracting Key Definitions In A Separate Map

A writer or information architect may want to store all key definitions in a separate map, away from the master map that produces content. This provides a layer of abstraction between, for example, a content writer and a localization expert who translates UI labels and other resources.

Key definitions made in maps are visible from other maps including referencing maps at a higher, lower, or the same level in the map hierarchy.

Keyref.ditamap

```
<map>
<!-- keys defined in this map -->
<title>Keyrefs Map</title>
<keydef keys="waterbirds" href="waterbirds.dita" processing-role="normal"/>
</map>
```

MainMap.ditamap

```
<map>
<title>Bird Guide</title>
<!-- Include map that defines keys -->
<topicref href="Keyref.ditamap" format="ditamap"/>
<topicref href="seabirds.dita"/>
</map>
```

Complete code for this example can be found in the Keyref__NestedMaps.zip file.