

Stage 3 proposal: Feature #13004

Contents

Stage 3 proposal: Feature #13004..... 3

 Finding Effective Key Definitions for a Key Scope..... 15

 Examples of keys..... 17

 The keyscope attribute..... 19

Stage 3 proposal: Feature #13004

Proposal to provide a mechanism for different effective key definitions at different locations within a map structure.

Champion

Chris Nitchie, Oberon Technologies, chris.nitchie@oberontech.com

Tracking information

Event	Date	Links
Stage 1 proposal accepted	June 21, 2011	Meeting Minutes
Stage 2 proposal submitted	Initial Revision: January 22, 2012 Final Revision: April 23, 2013	Initial Revision: HTML (no DITA by accident) Final Revision: HTML , DITA
Stage 2 proposal discussed	January 24, 2012 April 9, 2013 April 23, 2013	January 24, 2012 April 9, 2013 April 23, 2013
Stage 2 proposal approved	April 30, 2013	Minutes
Stage 3 proposal submitted to reviewers	July 8, 2013	David Helfinstine, Eliot Kimber
Stage 3 proposal (this document) submitted		

Approved technical requirements

Future post DITA 1.2 work from Issue #12007: Item 2: Consider adding a scoping mechanism for use with key definitions and possibly other aspects of DITA based on the map hierarchy or the combined element hierarchy within a set of maps. Related topic: Consider allowing subsequent key definitions within the same map or within peer maps to override previous key definitions, probably as part of some hierarchal scope, and probably in a fashion that is the same as or similar to override mechanisms to be developed for other features such as ditaval.

Dependencies or interrelated proposals

13041, cross-deliverable linking, depends heavily on this proposal.

Modified DTDs

The `topicref-atts` entity and its relatives from `map.mod` will be modified with the new `@keyscope` attribute.

```
<!ENTITY % topicref-atts
    "collection-type
        (choice |
         family |
         sequence |
         unordered |
         -dita-use-conref-target)
        #IMPLIED
    type
```

```

        CDATA
        #IMPLIED
processing-role
    (normal |
     resource-only |
     -dita-use-conref-target)
    #IMPLIED
scope
    (external |
     local |
     peer |
     -dita-use-conref-target)
    #IMPLIED
locktitle
    (no |
     yes |
     -dita-use-conref-target)
    #IMPLIED
format
    CDATA
    #IMPLIED
linking
    (none |
     normal |
     sourceonly |
     targetonly |
     -dita-use-conref-target)
    #IMPLIED
toc
    (no |
     yes |
     -dita-use-conref-target)
    #IMPLIED
print
    (no |
     printonly |
     yes |
     -dita-use-conref-target)
    #IMPLIED
search
    (no |
     yes |
     -dita-use-conref-target)
    #IMPLIED
chunk
    CDATA
    #IMPLIED
keyscope
    CDATA
    #IMPLIED
"
>
<!ENTITY % topicref-atts-no-toc
    'collection-type
        (choice |
         family |
         sequence |
         unordered |
         -dita-use-conref-target)
        #IMPLIED
    type
        CDATA

```

```

                                #IMPLIED
processing-role
    (normal |
     resource-only |
     -dita-use-conref-target)
                                #IMPLIED
scope
    (external |
     local |
     peer |
     -dita-use-conref-target)
                                #IMPLIED
locktitle
    (no |
     yes |
     -dita-use-conref-target)
                                #IMPLIED
format
    CDATA
linking
                                #IMPLIED
    (none |
     normal |
     sourceonly |
     targetonly |
     -dita-use-conref-target)
                                #IMPLIED
toc
    (no |
     yes |
     -dita-use-conref-target)
                                "no"
print
    (no |
     printonly |
     yes |
     -dita-use-conref-target)
                                #IMPLIED
search
    (no |
     yes |
     -dita-use-conref-target)
                                #IMPLIED
chunk
    CDATA
keyscope
                                #IMPLIED
    CDATA
'                                #IMPLIED
>
<!ENTITY % topicref-atts-without-format
    "collection-type
    (choice |
     family |
     sequence |
     unordered |
     -dita-use-conref-target)
                                #IMPLIED
    type
    CDATA
                                #IMPLIED

```

```

processing-role
    (normal |
     resource-only |
     -dita-use-conref-target)
        #IMPLIED

scope
    (external |
     local |
     peer |
     -dita-use-conref-target)
        #IMPLIED

locktitle
    (no |
     yes |
     -dita-use-conref-target)
        #IMPLIED

linking
    (none |
     normal |
     sourceonly |
     targetonly |
     -dita-use-conref-target)
        #IMPLIED

toc
    (no |
     yes |
     -dita-use-conref-target)
        #IMPLIED

print
    (no |
     printonly |
     yes |
     -dita-use-conref-target)
        #IMPLIED

search
    (no |
     yes |
     -dita-use-conref-target)
        #IMPLIED

chunk
    CDATA
        #IMPLIED

keyscope
    CDATA
        #IMPLIED
"
>

```

learningDomain.mod

```

<!-- Attributes that are common to each topicref specialization in this
domain -->
<!ENTITY % learningDomain-topicref-atts-no-chunk
    "navtitle
        CDATA
            #IMPLIED
        href
        CDATA
            #IMPLIED
        keyref
        CDATA
            #IMPLIED
    "

```

```

keys
    CDATA
    #IMPLIED
keyscope
    CDATA
    #IMPLIED
query
    CDATA
    #IMPLIED
copy-to
    CDATA
    #IMPLIED
outputclass
    CDATA
    #IMPLIED
scope
    (external |
     local |
     peer |
     -dita-use-conref-target)
    #IMPLIED
processing-role
    (normal |
     resource-only |
     -dita-use-conref-target)
    #IMPLIED
linking
    (targetonly|
     sourceonly|
     normal|
     none |
     -dita-use-conref-target)
    #IMPLIED
locktitle
    (yes|
     no |
     -dita-use-conref-target)
    #IMPLIED
toc
    (yes|
     no |
     -dita-use-conref-target)
    #IMPLIED
print
    (yes|
     no |
     printonly |
     -dita-use-conref-target)
    #IMPLIED
search
    (yes|
     no |
     -dita-use-conref-target)
    #IMPLIED
%univ-atts;"
>

```

mapMod.xsd

```

<xs:attributeGroup name="topicref-atts">
  <xs:annotation>
    <xs:documentation>

```

```

    The %topicref-atts; parameter entity represents a group of
    attributes
    used in numerous map elements: map, topicref, relcolspec, relcell,
    topichead,
    and topicgroup. The set is similar to those documented in <xref
    href="topicref-atts-no-toc.xml">topicref-atts-no-toc</xref>
    but includes the <i><keyword>toc</keyword></i> attribute.
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="collection-type" type="collection-type.class"/>
  <xs:attribute name="type" type="xs:string"/>
  <xs:attribute name="processing-role" type="processing-role-att.class" />
  <xs:attribute name="scope" type="scope-att.class"/>
  <xs:attribute name="locktitle" type="yesno-att.class"/>
  <xs:attribute name="format" type="xs:string"/>
  <xs:attribute name="linking" type="linkingtypes.class"/>
  <xs:attribute name="toc" type="yesno-att.class"/>
  <xs:attribute name="print" type="print-att.class"/>
  <xs:attribute name="search" type="yesno-att.class"/>
  <xs:attribute name="chunk" type="xs:string"/>
  <xs:attribute name="keyscope" type="xs:string"/>
</xs:attributeGroup>

<!-- ... -->

<xs:attributeGroup name="topicref-atts-no-toc">
  <xs:annotation>
    <xs:documentation>
      The %topicref-atts-no-toc; parameter entity represents the set
      of attributes used in the
      <xref href="reltable.xml">reltable</xref>
      map
      element. The set is similar to those documented in
      <xref href="topicref-atts.xml">topicref-atts</xref>
      but
      for &lt;<keyword>reltable</keyword>&gt; the <i><keyword>toc</
      keyword></i> attribute
      defaults to <q>no</q>. For the other elements that use the
      %topicref-atts;
      group declaration, the <i><keyword>toc</keyword></i> attribute
      doesn't have
      a default; they can inherit their <i><keyword>toc</keyword></i>
      value from
      the nearest container, otherwise it functions upon output as if set
      to <q>yes</q>.
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="collection-type" type="collection-type.class"/>
  <xs:attribute name="type" type="xs:string"/>
  <xs:attribute name="processing-role" type="processing-role-att.class" />
  <xs:attribute name="scope" type="scope-att.class"/>
  <xs:attribute name="locktitle" type="yesno-att.class"/>
  <xs:attribute name="format" type="xs:string"/>
  <xs:attribute name="linking" type="linkingtypes.class"/>
  <xs:attribute name="toc" type="yesno-att.class" default="no"/>
  <xs:attribute name="print" type="print-att.class"/>
  <xs:attribute name="search" type="yesno-att.class"/>
  <xs:attribute name="chunk" type="xs:string"/>
  <xs:attribute name="keyscope" type="xs:string"/>
</xs:attributeGroup>

```


Unlike the DTDs, in the schemas, the `topicref-atts-without-format` attribute group is defined in `mapGroupMod.xsd` instead of `mapMod.xsd`.

```
<xs:attributeGroup name="topicref-atts-without-format">
  <xs:annotation>
    <xs:documentation>
      The %topicref-atts-without-format; parameter entity represents a
      group of attributes
      used in numerous map elements: mapref,
      and topicgroup. The set is similar to those documented in <xref
      href="topicref-atts.xml">topicref-atts</xref> but
      excludes the <i><keyword>format</keyword></i> attribute.
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="collection-type" type="collection-type.class"/>
  <xs:attribute name="type" type="xs:string"/>
  <xs:attribute name="processing-role" type="processing-role-att.class" />
  <xs:attribute name="scope" type="scope-att.class"/>
  <xs:attribute name="locktitle" type="yesno-att.class"/>
  <xs:attribute name="linking" type="linkingtypes.class"/>
  <xs:attribute name="toc" type="yesno-att.class"/>
  <xs:attribute name="print" type="print-att.class"/>
  <xs:attribute name="search" type="yesno-att.class"/>
  <xs:attribute name="chunk" type="xs:string"/>
  <xs:attribute name="keyscope" type="xs:string"/>
</xs:attributeGroup>
```

`learningDomain.xsd`

```
<xs:attributeGroup name="learningDomain-topicref-atts-no-chunk">
  <xs:attribute name="navtitle" type="xs:string"/>
  <xs:attribute name="href" type="xs:string"/>
  <xs:attribute name="keyref" type="xs:string"/>
  <xs:attribute name="keys" type="xs:string"/>
  <xs:attribute name="keyscope" type="xs:string"/>
  <xs:attribute name="query" type="xs:string"/>
  <xs:attribute name="copy-to" type="xs:string"/>
  <xs:attribute name="outputclass" type="xs:string"/>
  <xs:attribute name="scope" type="scope-att.class"/>
  <xs:attribute name="processing-role" type="processing-role-
att.class" />
  <xs:attribute name="locktitle" type="yesno-att.class"/>
  <xs:attribute name="linking" type="linkingtypes.class"/>
  <xs:attribute name="toc" type="yesno-att.class"/>
  <xs:attribute name="print" type="print-att.class"/>
  <xs:attribute name="search" type="yesno-att.class"/>
  <xs:attributeGroup ref="univ-atts" />
</xs:attributeGroup>
```

Modified specification documentation

Provide a listing of new or modified topics. For existing topics, provide the location of the DITA 1.2 topic; for new topics, suggest where you think the topic should be located in the TOC.

List the exact language suggested for the topics. For existing topics, you can use a table to compare the DITA 1.2 text and the proposed text for DITA 1.3. For new topics, use the templates nested within this topic.

Table 1: dita-terminology.dita

DITA 1.2 Language		DITA 1.3 Language	
key definition	A <topicref> element or specialization of a <topicref> element that specifies the @keys attribute and defines one or more key names.	key definition	A <topicref> element or specialization of a <topicref> element that specifies the @keys attribute and defines one or more key names.
key resolution context	The root map that establishes the context for resolving key references. For a given key-resolution instance, there is at most one root map that defines the effective value of the key space, as determined by the key definition precedence rules..	key scope	A root map or section of a map hierarchy associated with a unique key space for purposes of key reference resolution.
key space	The effective set of unique key names that are defined by a given key resolution context. Within a given key resolution context, a key name has at most one binding.	key space	The set of unique key names that are defined for a given key scope. Within a given key scope, a key name has at most one definition, as determined by the key definition precedence rules.
		key resolution context	The key scope that establishes the context for resolving key references. For a given key-resolution instance, there is at most one key scope that defines the effective key space, which in turn provides the effective key definition.

Table 2: definition-of-ditamaps.dita

DITA 1.2 Language	DITA 1.3 Language
DITA maps are documents that organize topics and other resources into structured collections of information. DITA maps specify hierarchy and the relationships among the topics; they also provide the context in which keys are defined and resolved. DITA maps should have .ditamap file extensions.	DITA maps are documents that organize topics and other resources into structured collections of information. DITA maps specify hierarchy and the relationships among the topics; they also provide the contexts in which keys are defined and resolved. DITA maps should have .ditamap file extensions.
DITA maps also define keys and provide the context in which key references are resolved. A <topicref> element (or specialized <topicref> such as <keydef>) may be used to define a key which binds that key name to a specified resource.	<div>Disposition: / Status: One-word change: 'context' to 'contexts'</div> DITA maps also define keys and organize the contexts in which key references are resolved. A <topicref> or <map> element, or any specialization, may be used to create a new context for key definition and resolution. A <topicref> element, or specialized <topicref> such as <keydef>, may also be used to define a key which binds that key name to a specified resource for references in the same key scope.

Table 3: purpose-of-ditamaps.dita


DITA 1.2 Language		DITA 1.3 Language	
Defining keys	<p>Maps can define keys, which provide an indirect addressing mechanism that enhances portability of content. The keys are defined by <topicref> elements or specialization of <topicref> elements, such as <keydef>. The <keydef> element is a convenience element; it is a specialized type of a <topicref> element with the following attributes:</p> <ul style="list-style-type: none"> • A required @keys attribute • A @processing-role attribute with a default value of "resource-only". <p>Maps also are the context for resolving key-based references, such as elements that specify the @keyref or @conkeyref attribute.</p>	Defining keys and key scopes	<p>Maps can define keys, which provide an indirect addressing mechanism that enhances portability of content. The keys are defined by <topicref> elements or specialization of <topicref> elements, such as <keydef>. The <keydef> element is a convenience element; it is a specialized type of a <topicref> element with the following attributes:</p> <ul style="list-style-type: none"> • A required @keys attribute • A @processing-role attribute with a default value of "resource-only". <p>Maps also define the context or contexts for resolving key-based references, such as elements that specify the @keyref or @conkeyref attribute. Elements within a map structure that specify a @keyscope attribute create a new context for key reference resolution. Key references within such elements are resolved against the set of effective key definitions for that scope.</p>


Table 4: ditamap-attributes.dita







DITA 1.2 Language		DITA 1.3 Language	
<p>Disposition: / Status: The keyscope entry will ultimately go next to the entry for keys, which is currently missing from the spec. Reported as a doc fix.</p>		keyscope	<p>Defines a new scope for key definition and resolution, and gives the scope one or more names, separated by white space. Key definitions within a scope can only be addressed directly by key references within</p>

DITA 1.2 Language	DITA 1.3 Language
	the same scope. The key space defined by a key scope also inherits all of the key definitions from its parent scope, and if a key is defined in both the parent and child scopes, the definition from the parent scope takes precedence. Scoped key definitions are also added to the parent scope's key space, prepended by the scope name followed by a period. If a scope has more than one name, then each key within the scope will be contributed to the parent scope once for each prefix.

Table 5: overview_of_keys.dita

DITA 1.2 Language	DITA 1.3 Language
<p>To use key references, one must understand how keys are defined and bound to resources, how a map hierarchy establishes a key space, and the interaction of keys with conditional processing.</p> <p><i>New section: Key scopes</i></p>	<p>To use key references, one must understand how keys are defined and bound to resources, how a map hierarchy and key scopes establish key spaces, and the interaction of keys with conditional processing.</p> <p>Key scopes</p> <p>A key scope defines a key space associated with a segment of the map hierarchy, the contents of which are described below. It also defines a key reference resolution context for key references occurring within the scope-defining element, associated with that key space. That is, key references within a key scope are resolved against the key space associated with that key scope.</p> <p>A key scope is declared by setting the @keyscope attribute on a <map>, <topicref>, or any specialization of <map> or <topicref>, within the map structure. The value of the @keyscope attribute is the whitespace-delimited list of names for the key scope. The legal characters for a key scope name are the same as those for keys.</p> <p> Note:</p> <p>If the @keyscope attribute is specified both on a map element and on a ltopicref element referencing that map, only one new scope is introduced, not two. If the topicref has a scope of "local", the set of names for the resulting key scope is the union of the unique values of</p>

DITA 1.2 Language	DITA 1.3 Language
	<p>the two <code>@keyscope</code> attributes. If the <code>topicref</code> has a scope of "peer", only the scope names on the <code>topicref</code> apply, because there may be cases where the referenced map is unavailable.</p> <p>The key space associated with a key scope consists of the following, in order of precedence:</p> <ol style="list-style-type: none"> 1. All keys in the parent key scope's key space, if any. 2. All keys defined within the scope-defining element, including that element itself and any directly addressed, local scope maps contained by the scope-defining element, but <i>excluding</i> keys defined in other key scopes contained by the scope-defining element (i.e. child scopes). 3. Keys defined in child scopes, prefixed with the name of the key scope, followed by a period. If a child key scope has multiple names, then each of its keys is addressable from the parent key scope using any of the child key scope's names as a prefix. <p> Note: Since a parent scope contains the scope-qualified keys from its child scopes, and a child key scope inherits all key definitions from its parent scope, a child scope also contains all of the qualified key names from itself, as well as its "sibling" scopes. See Finding Effective Key Definitions for a Key Scope on page 15 for an example of this.</p> <p>For purposes of key definition precedence within a scope, the implicit qualified key definitions a parent key scope derives from a child key scope are considered to occur at the location of the child scope-defining element.</p> <p>The root element in the root map in a map hierarchy always necessarily defines a key scope, even if its <code>@keyscope</code> attribute is unspecified.</p>
<p>Key spaces</p> <p>A root map and its directly addressed, local scope descendant maps establish a unique key space within which each unique key name has exactly one binding to a set of resources.</p> <p>For the purposes of determining the effective key definitions for the key space represented by a given root map, a map tree is determined by considering only directly addressed, local scope maps descending from the root map. The order of subordinate maps is determined by the document order of the <code>topicrefs</code> that point to them. Indirect references to maps with key references are necessarily ignored until after the key space is determined.</p> <p>Maps addressed by <code><navref></code> do not contribute to the key space of a map tree. Maps referenced by <code><navref></code> are</p>	<p>Key spaces</p> <p>A key scope-defining element and its directly addressed, local scope descendant maps establish a unique key space within which each unique key name has exactly one binding to a set of resources.</p> <p>For the purposes of determining the effective key definitions for the key space represented by a given key scope, a map tree is determined by considering only directly addressed, local scope maps descending from the key scope-defining element. The order of subordinate maps is determined by the document order of the <code>topicrefs</code> that point to them. Indirect references to maps with key references are necessarily ignored until after the key space is determined.</p> <p>Maps addressed by <code><navref></code> do not contribute to the key space of a key scope. Maps referenced by <code><navref></code> are</p>

DITA 1.2 Language	DITA 1.3 Language
<p>equivalent to maps referenced with a scope of "peer" or "external" and therefore need not be present or available at the time the referencing map is processed for purposes of key space construction.</p>	<p>equivalent to maps referenced with a scope of "peer" or "external" and therefore need not be present or available at the time the referencing map is processed for purposes of key space construction.</p>
<p>Effective key definitions</p> <p>For a given key there is at most one effective definition within a key space. A key definition is the effective definition for a given key if it is the first, in document order, within the map document that contains it, and is the first in the map tree in breadth-first order. It is not an error for the same key name to be defined more than once within a map or map tree, and duplicate key definitions should be ignored without warning.</p> <p> Note: A given <topicref> element that defines more than one key may be the effective definition for some of its keys but not for others. It is the duplicate binding of a key name to its definition that is ignored, not the key-defining topic reference as a whole.</p> <p>Key definitions are not scoped by the map document within which they occur or by the element hierarchy of their containing map document. Keys do not have to be declared before they are referenced. The key space is effective for the entire document, so the order of key definitions and key references relative to one another within the map hierarchy is not significant, and keys defined in any map in the map tree are available for use with key references from all maps and topics processed in the context of the root map.</p> <p> Note: These rules mean that key definitions higher in the map tree hierarchy take precedence over key definitions lower in the map tree and that key definitions in referencing maps always take precedence over key definitions in referenced maps. These rules also mean that the entire key space must be determined before any keys can be resolved to their ultimately-addressed resources (if any).</p> <p> Note: Because keys are defined in maps, all key-based processing must be done in the context of a root map that establishes the effective key space.</p> <p>For key definitions in a submap to be included in the key space, there must be a direct URI reference to that submap from another directly addressed map in the map tree. However, if that same submap is referenced indirectly and has no direct URI reference as a backup (using @keyref without providing a fallback @href value, or using @conkeyref without providing a fallback @conref value), that reference is ignored for purposes</p>	<p>Effective key definitions</p> <p>For a given key there is at most one effective definition within a key space. A key definition is the effective definition for a given key if it is the first, in document order, within the key scope-defining element that contains it, and is the first in the map tree beneath the scope-defining element in breadth-first order. It is not an error for the same key name to be defined more than once within a map or key scope, and duplicate key definitions should be ignored without warning.</p> <p> Note: A given <topicref> element that defines more than one key may be the effective definition for some of its keys but not for others. It is the duplicate binding of a key name to its definition that is ignored, not the key-defining topic reference as a whole.</p> <p>Key definitions are scoped by the key scope-defining elements within which they occur. Keys do not have to be declared before they are referenced. The key space is effective for the entire key scope, so the order of key definitions and key references relative to one another within the hierarchy of a key scope is not significant, and keys defined in any map in the key scope are available for use with key references from all maps and topics processed in the context of the key scope.</p> <p> Note: These rules mean that key definitions higher in the map tree hierarchy take precedence over key definitions lower in the map tree and that key definitions in referencing maps always take precedence over key definitions in referenced maps. These rules also mean that all key spaces must be fully determined before any keys can be resolved to their ultimately-addressed resources (if any).</p> <p> Note: Because keys are defined in maps, all key-based processing must be done in the context of a root map that establishes the root key scope and scope hierarchy, or a specific scope within such a map.</p> <p>For key definitions in a submap to be included in the key space, there must be a direct URI reference to that submap from another directly addressed map in the map tree. However, if that same submap is referenced indirectly and has no direct URI reference as a backup (using @keyref without providing a fallback @href value, or using @conkeyref without providing a fallback</p>

DITA 1.2 Language	DITA 1.3 Language
of constructing the key space, and the definitions in that submap consequently do not enter into the construction of the key space at that point.	@conref value), that reference is ignored for purposes of constructing the key space, and the definitions in that submap consequently do not enter into the construction of the key space at that point.

Table 6: commonLRdefs.dita

DITA 1.2 Language	DITA 1.3 Language
topicref-atts attribute group (collection-type, processing-role, type, scope, locktitle, format, linking, toc, print, search, chunk)	topicref-atts attribute group (collection-type, processing-role, type, scope, locktitle, format, linking, toc, print, search, chunk, keyscope)
topicref-atts-no-toc attribute group (collection-type, processing-role, type, scope, locktitle, format, linking, toc, print, search, chunk)	topicref-atts-no-toc attribute group (collection-type, processing-role, type, scope, locktitle, format, linking, toc, print, search, chunk, keyscope)
topicref-atts-without-format attribute group (collection-type, processing-role, type, scope, locktitle, linking, toc, print, search, chunk)	topicref-atts-without-format attribute group (collection-type, processing-role, type, scope, locktitle, linking, toc, print, search, chunk, keyscope)
<i>New attribute table entry for 'keyscope', to be referenced from topicref-atts.dita</i>	Specifies that the element marks the boundaries of a key scope. See The keyscope attribute on page 19 for details on how to use the keyscope attribute.

Table 7: thekeysattribute.dita

DITA 1.2 Language	DITA 1.3 Language
A keys attribute consists of one or more space-separated keys. Map authors define keys using a topicref or topicref specialization that contains the “keys” attribute. Each key definition introduces a global identifier for a resource referenced from a map. Keys resolve to the resources given as the href value on the key definition topicref element, to content contained within the key definition topicref element, or both.	A keys attribute consists of one or more space-separated keys. Map authors define keys using a topicref or topicref specialization that contains the “keys” attribute. Each key definition introduces an identifier for a resource referenced from the map. Keys resolve to the resources given as the href value on the key definition topicref element, to content contained within the key definition topicref element, or both.
	Disposition: / Status: One-word change: removed the word 'global' from, 'Each key definition introduces a global identifier...'.

Finding Effective Key Definitions for a Key Scope

When searching for the effective definition for a given key name from within a scope, check each scope between the root scope and the referencing scope, inclusive, starting from the root. The first matching definition - including implicit scope-qualified key names contributed by child key scopes - should be used to resolve the reference.

Consider the following map:

```
<map>
  <title>Root Map</title>

  <keydef keys="a"/>

  <topicgroup keyscope="A">
    <keydef keys="b"/>
```

```

<topicgroup keyscope="A-1">
  <keydef keys="c"/>
</topicgroup>

<topicgroup keyscope="A-2">
  <keydef keys="d"/>
</topicgroup>
</topicgroup> <!-- End scope A -->

<topicgroup keyscope="B">
  <keydef keys="a"/>
  <keydef keys="e"/>

  <topicgroup keyscope="B-1">
    <keydef keys="f"/>
  </topicgroup>

  <topicgroup keyscope="B-2">
    <keydef keys="g"/>
  </topicgroup>
</topicgroup> <!-- End scope B -->
</map>

```

The key scopes in this map form a tree structure.

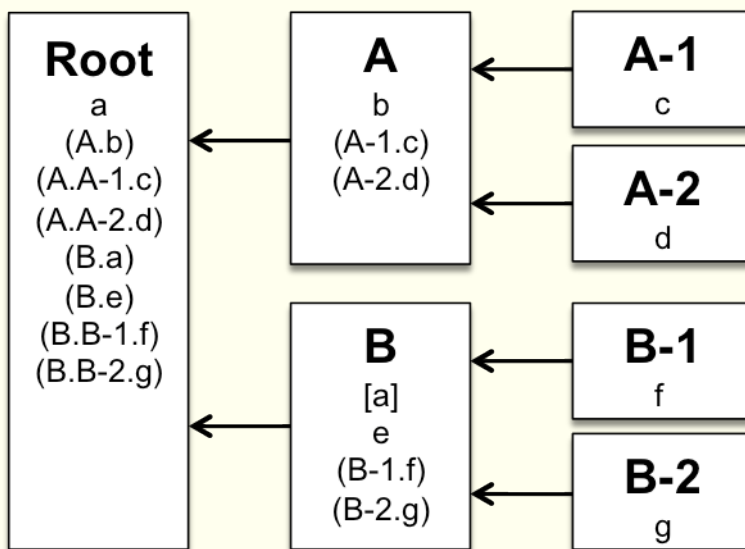


Figure 1: Sample Key Scope Structure

- Explicit key definitions are shown undecorated.
- Implicit qualified key names contributed by child scopes are shown in parentheses.
- Keys overridden by their parent scopes are in square brackets.
- The arrows denote parentage of each scope.

Here are some examples of various potential key references in various scopes from the above example.

- **Within Scope A-2**

- `keyref="a"` resolves to the definition for "a" at the root scope.
- `keyref="d"` resolves to the local definition.
- `keyref="A-2.d"` also resolves to the local definition, because that qualified key name is available in the parent scope.
- `keyref="c"` does not resolve, because no definition for that key exists in any of the ancestor scopes.

- However, `keyref="A-1.c"` does resolve, because that definition is available in the parent scope.
- As does `keyref="A.A-1.c"`, because that definition is available at the root scope.
- This scope can also reference key definitions within the "B" scope descending from the root, such as `keyref="B.e"` or `keyref="B.B-2.g"`, due to the presence of implicit qualified names for those keys at the root scope.
- **Within Scope B**
 - `keyref="e"` resolves to the local definition.
 - `keyref="a"` resolves to the *root-level* definition of "a", *not* the local definition. Keys from the parent scope override any local definitions.
 - However, `keyref="B.a"` resolves to the *local* definition of "a", as inherited from the root scope.
 - `keyref="g"` does not resolve, because no definition for that key exists in this or any parent scope.
 - However, `keyref="B-2.g"` will resolve due to the availability of the implicit qualified definition in the current scope.

Examples of keys

Disposition: / Status:

This topic already exists. However, since the changes to this topic are strictly additive - new sections with new examples - I've written up the new sections here instead of using the tabular format normally used for updates to existing topics.

Topic reuse with varying key behavior

In this example, the “oil change” procedures for Tractor X and Tractor Y are similar enough to reuse the same topic for both, but the name of the product is included in the title via `keyref`. In the map below, the topic `common/oilchange.dita` is referenced twice. Each reference is wrapped within a `topicgroup` that defines a key scope, and each scope has its own definition of the `ProductName` key.

```
<map xml:lang="en">
  <title>Equipment Oil Change Procedures</title>

  <topicgroup keyscope="Tractor-X">
    <keydef keys="ProductName">
      <topicmeta>
        <linktext>Tractor X</linktext>
      </topicmeta>
    </keydef>
    <topicref href="common/oilchange.dita"/>
  </topicgroup>

  <topicgroup keyscope="Tractor-Y">
    <keydef keys="ProductName">
      <topicmeta>
        <linktext>Tractor Y</linktext>
      </topicmeta>
    </keydef>
    <topicref href="common/oilchange.dita"/>
  </topicgroup>
</map>

<task id="oilchange" xml:lang="en">
  <title>Changing the Oil on the <ph keyref="ProductName"/></title>
  <!-- etc. -->
</task>
```

Combination of standalone maps with their own key spaces

```
<map xml:lang="en">
  <title>Training Courses</title>
  <mapref href="course-1.ditamap" keyscope="course-1"/>
  <mapref href="course-2.ditamap" keyscope="course-2"/>
  <mapref href="course-3.ditamap" keyscope="course-3"/>
  <topicref keys="summary" href="omnibus-summary.dita"/>
</map>
```

This map combines three standalone maps, each with its own set of keys. Each of the `mapref` elements is marked with a `keyscope` attribute, meaning that the key definitions within the referenced maps will be confined to those sub-structures.

Since the `summary` key is defined in this map as part of the implicit root scope, and keys defined in parent scopes override those in child scopes, its definition overrides any conflicting definition of `summary` in any of the referenced maps.

Examples of scope-qualified key references

```
<map xml:lang="en">
  <title>Scoped Key Example</title>
  <topicgroup keyscope="scope-1">
    <keydef keys="key-1" href="topic-1.dita" />

    <!-- This topicref will refer to topic-1.dita.
         This key is defined in this scope. -->
    <topicref keyref="key-1" />

    <!-- This topicref will also refer to topic-1.dita.
         This key is inherited from the root scope. -->
    <topicref keyref="scope-1.key-1" />

    <!-- This topicref will refer to topic-2.dita.
         This key is inherited from the root scope. -->
    <topicref keyref="scope-2.key-1" />
  </topicgroup>

  <topicgroup keyscope="scope-2">
    <keydef keys="key-1" href="topic-2.dita" />

    <!-- This topicref will refer to topic-2.dita.
         This key is defined in this scope. -->
    <topicref keyref="key-1" />

    <!-- This topicref will refer to topic-1.dita.
         This key is inherited from the root scope. -->
    <topicref keyref="scope-1.key-1" />

    <!-- This topicref will refer to topic-2.dita.
         This key is inherited from the root scope. -->
    <topicref keyref="scope-2.key-1" />
  </topicgroup>

  <!-- This topicref will result in an error, because
       no key definition 'key-1' exists at this scope. -->
  <topicref keyref="key-1" />

  <!-- This topicref will refer to topic-1.dita.
       Keys defined in a scope become a part of the
       parent scope, prefixed with the scope name and
       a period. -->
```

```

<topicref keyref="scope-1.key-1" />

<!-- This topicref will refer to topic-2.dita. -->
<topicref keyref="scope-2.key-1" />

</map>

```

This example illustrates the way keys defined within a key scope can be addressed from key references outside that scope.

The keyscope attribute

The keyscope attribute consists of one or more space-separated key scope names. Map authors define the boundaries for key scopes by specifying the keyscope attribute on map elements, topicref elements, or elements that are specializations of map or topicref. Such elements, their contents, and any locally-scoped content referenced from within the element, are considered to be part of the scope. Keys defined within a scope are only directly referenceable from within the same scope. They can be referenced from the parent scope using the scope's name, followed by a period, followed by the key name.



Note: Within a given root map, two distinct key scopes with the same name have no relationship with each other aside from that implied by their relative locations in the key scope hierarchy. They do not, for example, share key definitions. There is no such thing as a non-contiguous key scope. The only processing impact of a key scope's names is in defining the prefixes used when contributing qualified key names to the parent scope.