Stage 3 proposal: Feature #105 Redesign Chunking

Simplify how the `@chunk` attribute is defined to 1) make it easier for authors to use, and 2) make implementation easier and more reliable.

Champion

Provide information about the champion. If the proposal is submitted by a subcommittee, include the name of the point person. He or she should have prepared this proposal and thoroughly understand all of the content. The point person must be present at the TC calls when this proposal is discussed.

Tracking information

Event	Date	Links
Stage 1 proposal accepted	13 March 2018	Minutes
Stage 2 proposal submitted	Submitted 26 March 2018, updated 9 April 2018	DITA source, HTML
Stage 2 proposal discussed	3 April and 10 April	Minutes for 3 April 2018, 10 April 2018
Stage 2 proposal approved	17 April	Minutes
Stage 3 proposal submitted to reviewers	December 2018	Stan Doherty, Eliot Kimber (extra review from Chris Nitchie)
Stage 3 proposal (this document) submitted to TC	22 April 2019	SVN link

Approved technical requirements

- No change to grammar files. The attribute name is the same; chunk tokens are not defined in the grammar for 1.x and this will not change in 2.x.
- For some processors or output formats, chunking is irrelevant. Where it is irrelevant, nothing changes for processors.
- All previously defined tokens for `@chunk` will be removed from the specification.
- Two new tokens will be defined in the specification:

1. `chunk="combine"` This is roughly equivalent to `chunk="to-content"` in DITA 1.x.
   - When specified on a map, everything within the scope of that map is published as a single result document.
   - When specified on a branch of a map (or reference to a map), everything within the scope of that branch or reference is published as a single result document.
   - There is no way to undo the combine action for something within that scope.
2. `chunk="split"` This is roughly equivalent to `chunk="by-topic"` in DITA 1.x.
   - When specified on a `<topicref>`, it indicates that all topics *within the referenced document* should be split into multiple documents. ***For example***, in a context where each individual DITA document is

published as a single HTML file, specifying `chunk="split"` on a reference to a document that contains five topics will result in 5 documents + 5 output files.
- When a document is split, the effective map hierarchy remains the same. ***In the example from the previous item***, this means that evaluating `@chunk` for the one map reference would result in five references, which preserve any nesting hierarchy from that document.
- When specified on a root map, it indicates that all references should treat `chunk="split"` as the default operation. Setting `chunk="combine"` anywhere will override that default for that section of the map.
- When specified on a nested map, it indicates that the all references within the scope of that nested map should treat `chunk="split"` as the default operation. Setting `chunk="combine"` anywhere will override that default for that section of the map.
- Specifying `chunk="split"` on any other container element that does not reference a document (such as on `<topicgroup>`) has no meaning.

3. While we view this use as unlikely, the attribute is left open for application-specific tokens.

Dependencies or interrelated proposals

N/A

Modified grammar files

N/A

Modified terminology

N/A

Modified specification documentation

The following section of the specification should be removed, and replaced with the updated topics attached below:

- 2.4.5 Chunking
  - 2.4.5.1 Using the `@chunk` attribute
  - 2.4.5.2 Chunking examples

The current definition of `@chunk` in the Attributes common to many map elements is general enough that it applies equally to DITA 1.3 and DITA 2.0.

Migration plans for backwards incompatibilities

The best way to address this change is a search/replace type operation that finds values in the `@chunk` attribute. When the value contains `to-content`, replace the entire value with `combine`. When the value contains `by-topic`, replace the entire value with `split`.

Chunking

Content often needs to be delivered in a different granularity than it is authored. The `@chunk` attribute enables map authors to specify that multiple source documents should be combined into a single document for delivery, or that a single source document should be split into multiple documents for delivery.

About the `@chunk` attribute

The `@chunk` attribute is intended to handle cases where the best organization for creating DITA topics is not equivalent to the best organization for publishing those topics.

The `@chunk` attribute is composed of a single token without any white space. DITA defines two tokens for the `@chunk` attribute, `combine` and `split`. Other tokens may be defined by applications but support for those tokens will vary.

chunk="combine"	The `combine` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each source XML document. When some or all of those source XML documents are better presented as a single output artifact, setting `chunk="combine"` instructs a processor to combine the referenced source documents for rendering purposes.
chunk="split"	The `split` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each single source XML document, regardless of how many DITA topics exist within each source document. When a source XML document containing many topics is better rendered as multiple output artifacts, setting `chunk="split"` instructs a processor to split each topic from the referenced source document into its own document for rendering purposes.

The following rules apply to all values of the `@chunk` attribute:

- The `@chunk` attribute describes how a processor can split or combine source DITA documents into alternate organizational schemes for rendering purposes. This means that the `@chunk` attribute is only relevant when the organization of source DITA documents has an effect on organization of published documents. When the source document organization has no effect on published output, such as when producing a single PDF or EPUB, implementations *MAY* ignore the `@chunk` attribute.
- When the `@chunk` attribute results in more or fewer documents based on the `combine` or `split` tokens, the hierarchy of topics within the resulting map and topic organization *SHOULD* match the hierarchy in the original topics and maps.
- When the `@chunk` attribute results in more or fewer documents, implementations *MAY* create their own naming schemes for those reorganized documents.
- The `@chunk` attribute does not cascade to nested `<topicref>` elements or to nested map references.
- `@chunk` attribute values apply to DITA topic documents referenced from a map. Processors *MAY* apply equivalent processing to non-DITA documents.

Using the `@chunk` attribute to combine documents

The `@chunk` attribute can be set to `chunk="combine"` to produce a single result document out of topic documents that are best managed independently.

The `combine` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each source XML document. When some or all of those source XML documents are better presented as a single output artifact, setting `chunk="combine"` instructs a processor to combine the referenced source documents for rendering purposes.

Processing `chunk="combine"`

- When specified on a map, all source DITA documents referenced by the map are treated as one DITA document.
- When specified on a branch of a map, all source DITA documents referenced within that branch are treated as one DITA document.

  Note: This is true regardless of whether the element that specifies `@chunk` refers to a topic or specifies a heading. In cases such as `<topicgroup>` where a grouping element specifies `chunk="combine"`, the equivalent DITA document would be a single DITA document with a `<dita>` root element grouping peer topics.
- When specified on a reference to a map, all source DITA documents within the scope of the referenced map are treated as one DITA document.
- Once `chunk="combine"` is specified on a map, branch, or map reference, all source DITA documents grouped by that reference are treated as a single resource. Any additional `@chunk` attributes on elements within the hierarchy are ignored.

Using the `@chunk` attribute to split documents

The `@chunk` attribute can be set to `chunk="split"` to produce multiple result documents out of topics that are best managed within a single source document.

The `split` token in `@chunk` is intended for cases where a publishing process normally results in a single output artifact for each single source XML document, regardless of how many DITA topics exist within each source document. When a source XML document containing many topics is better rendered as multiple output artifacts, setting `chunk="split"` instructs a processor to split each topic from the referenced source document into its own document for rendering purposes.

Processing `chunk="split"`

- When specified on a `<topicref>` element that refers to a source DITA document, it indicates that all topics within the referenced document should be rendered as individual documents.
- When specified on an element such as `<topicgroup>` that does not refer to a topic or result in a published topic, the attribute has no meaning.
- When specified on a map, `chunk="split"` sets a default operation for all source DITA documents in the map (outside the context of relationship tables). The default `split` value is used except where a `combine` value is encountered, in which case `combine` takes over for that entire branch.

Using the `@chunk` attribute for other purposes

Additional tokens can be defined by applications for use in the `@chunk` attribute. These tokens are necessarily implementation dependent and might not be supported by other applications.

Impact of `@chunk` on linking

When documents split or combine during processing, links to the original documents will need to be adjusted. Managing these links will generally be more straightforward with keys, but even with keys the intended link target might be difficult to determine.

Often there is only one logical way for links to a chunked document to resolve.

- For direct links using `@href` or indirect links using `@keyref`: when a topic is only rendered once within the scope of a map, and the rendered topic is either split into a new document or combined with other documents, links resolve to the single new instance. For example, if `document.dita` is only rendered once within a map, links to `document.dita#someTopic` can only resolve to that topic, even if the topic is combined with a larger branch or split into a new document.
- For indirect links using `@keyref`: even if a topic is rendered more than once within a map, keys can be used to disambiguate links. For example, if each rendered copy of `document.dita` has a unique key definition, links to each key can only resolve to that specific instance.

In other cases, the link resolution might be ambiguous.

- For direct links using `@href`: when a topic is rendered multiple times within the scope of a map, link resolution is ambiguous. (This is also true in many cases without chunking.) For example, if `document.dita` is rendered as a single document on its own, but also used with `chunk="split"` to create multiple result documents, the target of `href="document.dita#someTopic"` could resolve to either result document.
- For indirect links using `@keyref`: when a topic is rendered multiple times within the scope of a map, and the key defining element is **not** the same element that results in the rendered topic, the target of the link is ambiguous (as with the `@href` case). For example, if a key `sampleKey` is defined for `document.dita#someTopic` using `<keydef>`, but is only rendered by reference to the file or to that key, the target of `keyref="sampleKey"` is ambiguous.

When evaluating `@chunk` results in ambiguous links, those links can resolve to any appropriate instance. Implementations *MAY* provide alternate mechanisms to disambiguate the link, provided those mechanisms do not otherwise change how standard attributes like `@chunk` or `@keyref` are resolved. For example, if key scopes are used, an application might determine that the best resolution for `@href` direct links is to an instance of the topic within the same scope.

For one known edge case where a topic is rendered more than once, it is not possible to associate a key with a specific rendered instance of a topic, so it is impossible to remove all ambiguity.

- For indirect links using `@keyref`: when a document is split using `chunk="split"`, it is only possible for the element using `@chunk` to associate a key with the first or root topic in the document. Other key definition elements can be used to associate keys with other topics in the same document. However, if a *child* topic from the document is rendered based on both the `chunk="split"` context and based on another context, it is not possible to define a unique key for only the child topic that results from evaluating `chunk="split"`.

Examples of the `@chunk` attribute

These examples illustrate the processing expectations for various scenarios that involve the `@chunk` attribute. Processing examples use either before and after sample markup or expanded syntax that shows the equivalent markup without the `@chunk` attribute.

Note: Examples use sample files with modified file names to help illustrate equivalent before and after resolution of `@chunk` attributes. However, there is no requirement for implementations processing `@chunk` to generate files, as long as the rendered result is split or combined as described. If generating files, actual file names are implementation dependent.

Example: Using `@chunk` to combine all documents into one

Where a publishing system would typically render each topic document as an independent result document, a single `@chunk` attribute can be used to render all content as a single document.

Consider the following source documents, with root map `input.ditamap`:

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="background.dita">
    <!-- more background topics -->
  </topicref>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <!-- more topics -->
</map>

background.dita:
<topic id="background">
  <title>Prerequisite concepts</title>
  <shortdesc>This information is necessary before starting</shortdesc>
  <body><!-- ...background content... --></body>
</topic>

goals.dita:
<topic id="goals">
  <title>Lesson gals</title>
  <shortdesc>After you complete the lesson, ...</shortdesc>
  <body><!-- ...goal content... --></body>
</topic>
```

Figure 1: Input map without chunking

For many systems or output formats, each document in the map will be rendered as an independent document. For example, rendering this map as HTML5 might result in `background.html` and `goals.html` (along with other HTML5 files). In such cases, if output requirements demand only a single result document, adding `chunk="combine"` to the root map element instructs a processor to render one document that combines all topics.

```
input.ditamap:
<map chunk="combine">
  <title>Lesson plan</title>
```

```
  <topicref href="background.dita">
    <!-- more background topics -->
  </topicref>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <!-- more topics -->
</map>
```

Figure 2: Root map with chunking specified

The result of evaluating this `@chunk` attribute is equivalent to the following map and topic document; content from all topics within the map has now been combined into a single result, with the topic order and topic nesting structure matching the original map hierarchy.

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="input.dita"/>
</map>

input.dita:
<dita>
  <!-- original content of background.dita -->
  <topic id="background">
    <title>Prerequisite concepts</title>
    <shortdesc>This information is necessary before starting</shortdesc>
    <body><!-- ...background content... --></body>
    <!-- more background topics -->
  </topic>
  <!-- original content of goals.dita -->
  <topic id="goals">
    <title>Lesson gals</title>
    <shortdesc>After you complete the lesson, ...</shortdesc>
    <body><!-- ...goal content... --></body>
    <!-- more goal topics -->
  </topic>
  <!-- more topics -->
</dita>
```

Figure 3: Equivalent source content

Example: Using `@chunk` to render a document from one branch

Where a publishing system would typically render each topic document as an independent result document, a single `@chunk` attribute can be used to render individual branches of a map as single documents.

Consider the following source documents, with root map `input.ditamap`:

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita">
    <!-- more tasks in the first lesson -->
  </topicref>
  <topicref href="nextLesson.dita">
    <!-- more tasks in the next lesson -->
  </topicref>
  <!-- More branches -->
</map>
```

```
firstLesson.dita:
<task id="firstLesson">
  <title>Starting to work with scissors</title>
  <shortdesc>This lesson will teach ...</shortdesc>
  <taskbody><!-- ... --></taskbody>
</task>

nextLesson.dita:
<task id="nextLesson">
  <title>Advanced cutting</title>
  <shortdesc>This lesson will introduce complicated shapes...</shortdesc>
  <taskbody><!-- ... --></taskbody>
</task>
```

Figure 4: Input map without chunking

For many systems or output formats, each document in the map will be rendered as an independent document. For example, rendering this map as HTML5 might result in `goals.html`, `firstLesson.html`, and `nextLesson.html`, while child documents within each branch would each result in their own HTML files.

When output requirements demand some portions of the map be combined into a single document, adding `chunk="combine"` to a branch of the map instructs a processor to render one document that combines all topics in that branch. This is particularly useful when the topics need to be rendered independently for other contexts, or when the way topics are contributed makes creating a single source document impossible.

In the following sample, the original map is updated with `@chunk` attributes to indicate that each lesson branch should render as a single result document; topics in the first branch with `goals.dita` are not affected as a result of the `@chunk` attribute.

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita" chunk="combine">
    <!-- more tasks in the first lesson -->
  </topicref>
  <topicref href="nextLesson.dita" chunk="combine">
    <!-- more tasks in the next lesson -->
  </topicref>
  <!-- More branches -->
</map>
```

Figure 5: Map with chunking specified for one branch

The result of evaluating this `@chunk` attribute is equivalent to the following map and topic documents. Content from each combined branch has now been combined into a single result document for each branch, with the order and topic nesting structure matching the original map hierarchy. Content from outside of those branches remains unchanged.

```
input.ditamap:
<map>
  <title>Lesson plan</title>
  <topicref href="goals.dita">
    <!-- more goal topics -->
  </topicref>
  <topicref href="firstLesson.dita"/>
  <topicref href="nextLesson.dita"/>
  <!-- More branches -->
</map>
```

```
firstLesson.dita:
<task id="firstLesson">
  <title>Starting to work with scissors</title>
  <shortdesc>This lesson will teach ...</shortdesc>
  <taskbody><!-- ... --></taskbody>
  <!-- more tasks in the first lesson -->
</task>

nextLesson.dita:
<task id="nextLesson">
  <title>Advanced cutting</title>
  <shortdesc>This lesson will introduce complicated shapes...</shortdesc>
  <taskbody><!-- ... --></taskbody>
  <!-- more tasks in the next lesson -->
</task>
```

Figure 6: Equivalent source content

Example: Using `@chunk` to combine a group of topics

The `@chunk` attribute can be used on grouping elements to combine multiple source documents into one result document.

Assume the following map `input.ditamap`, where `@chunk` is used on both `<topicgroup>` and `<topichead>`.

```
<map>
  <title>Groups are combined</title>
  <topicgroup chunk="combine">
    <topicref href="ingroup1.dita"/>
    <topicref href="ingroup2.dita"/>
  </topicgroup>
  <topichead chunk="combine">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
    <topicref href="inhead1.dita"/>
    <topicref href="inhead2.dita"/>
  </topichead>
</map>
```

- The result of evaluating the `@chunk` attribute on `<topicgroup>` is equivalent to a single DITA document with the content of both `ingroup1.dita` and `ingroup2.dita`.
- The `@chunk` attribute on `<topichead>` also results in a single DITA document. In many applications, a `<topichead>` is equivalent to a single title-only topic; in that case, the chunked result is equivalent to a root topic with the title "Heading for a branch", containing as child topics the content of both `inhead1.dita` and `inhead2.dita`. If `<topichead>` is ignorable in the current processing context, the chunked result would be equivalent to processing `<topicgroup>` (a single DITA document with the content of both `inhead1.dita` and `inhead2.dita`).

```
<map>
  <title>Groups are combined</title>
  <topicref href="chunkgroup-1.dita"/>
  <topicref href="chunkgroup-2.dita"/>
</map>

chunkgroup-1.dita
<dita>
  <!-- content of ingroup1.dita -->
  <!-- content of ingroup2.dita -->
</dita>

chunkgroup-2.dita
<dita>
```

```
  <topic id="head">
    <title>Heading for a branch</title>
    <!-- content of inhead1.dita -->
    <!-- content of inhead2.dita -->
  </topic>
</dita>
```

Figure 7: Equivalent source content

Example: Using `@chunk` to combine nested documents

Special attention is necessary when combining a nested map hierarchy that includes documents with their own nested topics.

Consider the following source map `input.ditamap`:

```
input.ditamap:
<map chunk="combine">
  <title>Generation example</title>
  <topicref href="ancestor.dita">
    <topicref href="middle.dita">
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

Figure 8: Input map without chunking

In this case, the `@chunk` attribute instructs a processor to treat the three topics as a single combined DITA document, while preserving the original map hierarchy. Now consider the following three topic documents, each of which includes nested or peer topics:

```
ancestor.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
  </topic>
</dita>

middle.dita:
<topic id="middle-root">
  <title>Root topic in middle doc</title>
  <body><!-- ... --></body>
  <topic id="middle-child">
    <title>Child of root topic in middle doc</title>
    <!-- body content, maybe more children of middle topic's root -->
  </topic>
</topic>

child.dita:
<topic id="child">
  <title>Small child topic</title>
  <!-- small child topic content -->
```

```
</topic>
```

Figure 9: Source documents with nested structures

When `chunk="combine"` is evaluated, the three source documents are combined into one. Both the ancestor and middle documents have child topics that must be taken into account.

- `ancestor.dita` has a root `<dita>` element, so content from each nested topic reference is located after any nested topics within the final child of the `<dita>` element.
- `middle.dita` does not have `<dita>` but does have a nested topic, so content from any nested topic references is located after that nested topic.

```
input.ditamap:
<map>
  <title>Generation example</title>
  <topicref href="input.dita"/>
</map>

input.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
    <!-- content of middle.dita combined here -->
    <topic id="middle-root">
      <title>Root topic in middle doc</title>
      <body><!-- ... --></body>
      <topic id="middle-child">
        <title>Child of root topic in middle doc</title>
        <!-- body content, maybe more children of middle topic's root -->
      </topic>
      <!-- content of child.dita combined here -->
      <topic id="child">
        <title>Small child topic</title>
        <!-- small child topic content -->
      </topic>
    </topic>
  </topic>
</dita>
```

Figure 10: Equivalent source content

Example: Using `@chunk` to split documents

When topics are most easily created or generated in a single DITA document, `chunk="split"` will instruct processors to render them individually when possible.

Splitting a single document in the map

Consider the following example, where a map includes generated topics used to document message numbers from an application:

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
```

```
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>

about.dita:
<topic id="about">
  <title>About this guide</title>
  <shortdesc>Warnings or errors will appear if...<shortdesc>
</topic>

messages-install.dita:
<dita>
  <topic id="INS001">
    <title>INS001: Installation failure</title>
    <!-- explanation and recovery... -->
  </topic>
  <!-- more install messages... -->
</dita>

messages-run.dita:
<dita>
  <topic id="RUN001">
    <title>RUN001: Failed to initialize</title>
    <!-- explanation and recovery... -->
  </topic>
  <!-- hundreds of messages... -->
  <topic id="RUN999">
    <title>RUN999: Out of memory</title>
    <!-- explanation and recovery... -->
  </topic>
</dita>

messages-other.dita:
<topic id="othermsg">
  <title>Other messages</title>
  <shortdesc>You could also encounter ...</shortdesc>
  <topic id="OTHER001">
    <title>OTHER001: Analyzer is tired</title>
    <!-- explanation and recovery... -->
  </topic>
  <topic id="OTHER002">
    <title>OTHER002: Analyzer needs to be updated</title>
    <!-- explanation and recovery... -->
  </topic>
</topic>
```

Figure 11: Source map and topics

In a normal build to HTML5, this map might result in four result documents `about.html`, `messages-install.html`, `messages-run.html`, and `messages-other.html`. With hundreds of messages in `messages-run.dita`, it will be better in some situations to render one result document for each message topic in the document. This is done by setting `chunk="split"` on the topic reference.

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita" chunk="split"/>
    <topicref href="messages-other.dita"/>
  </topicref>
```

```
</map>
```

Figure 12: Splitting all topics in one document

The result of evaluating `@chunk` in this case is equivalent to the following map and topics. While `messages-run.dita` is now split into hundreds of topics, other topics in the map are unaffected.

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="RUN001.dita"/>
    <!-- hundreds of messages... -->
    <topicref href="RUN999.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>

RUN001.dita:
<topic id="RUN001">
  <title>RUN001: Failed to initialize</title>
  <!-- explanation and recovery... -->
</topic>

RUN999.dita:
<topic id="RUN999">
  <title>RUN999: Out of memory</title>
  <!-- explanation and recovery... -->
</topic>
```

Figure 13: Equivalent source content

Note: Because the `@chunk` attribute does not cascade, even if the reference to `messages-install.dita` had child topic references, they would be unaffected by the `chunk="split"` value in this example.

Splitting every document in the map

Similarly, because setting `chunk="split"` on the map element sets a default for the entire map, the following change to the original map would result in *every* referenced DITA document being split into one document per topic. The only source document not affected by this split is `about.dita`, because it only contained a single topic to begin with.

```
<map chunk="split">
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="messages-install.dita"/>
    <topicref href="messages-run.dita"/>
    <topicref href="messages-other.dita"/>
  </topicref>
</map>
```

Figure 14: Splitting every topic in the map

Using `chunk="split"` on the map is equivalent to the following structure:

- `about.dita` is unchanged.
- `messages-install.dita` is split into one document per message (as in the previous example that split `messages-run.dita`).
- `messages-run.dita` is split exactly as in the previous example.

- `messages-other.dita` contains a root topic and two child topics, so it results in three documents. The hierarchy of those documents is preserved in the map.

```
<map>
  <title>Message guide for WidgetAnalyzer</title>
  <topicref href="about.dita">
    <topicref href="INS001.dita"/>
    <!-- more install messages... -->
    <topicref href="RUN001.dita"/>
    <!-- hundreds of messages... -->
    <topicref href="RUN999.dita"/>
    <topicref href="othermsg.dita">
      <topicref href="OTHER001.dita"/>
      <topicref href="OTHER002.dita"/>
    </topicref>
  </topicref>
</map>

INS001.dita:
<topic id="INS001">
  <title>INS001: Installation failure</title>
  <!-- explanation and recovery... -->
</topic>

RUN001.dita:
<topic id="RUN001">
  <title>RUN001: Failed to initialize</title>
  <!-- explanation and recovery... -->
</topic>

RUN999.dita:
<topic id="RUN999">
  <title>RUN999: Out of memory</title>
  <!-- explanation and recovery... -->
</topic>

othermsg.dita:
<topic id="othermsg">
  <title>Other messages</title>
  <shortdesc>You could also encounter ...</shortdesc>
</topic>

OTHER001.dita:
<topic id="OTHER001">
  <title>OTHER001: Analyzer is tired</title>
  <!-- explanation and recovery... -->
</topic>

OTHER002.dita:
<topic id="OTHER002">
  <title>OTHER002: Analyzer needs to be updated</title>
  <!-- explanation and recovery... -->
</topic>
```

Figure 15: Equivalent source content

Example: Using `@chunk` to split nested documents

Special attention is necessary when evaluating the map hierarchy that results from splitting a documents with their own nested topics.

Consider the following source map `input.ditamap`:

```
input.ditamap:
```

```
<map chunk="split">
  <title>Generation example</title>
  <topicref href="ancestor.dita">
    <topicref href="middle.dita">
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

Figure 16: Input map without chunking

In this case, the `@chunk` attribute instructs a processor to render every topic in each of the three documents as its own document, while preserving any hierarchy from those documents. Now consider the following three topic documents, each of which includes nested or peer topics:

```
ancestor.dita:
<dita>
  <topic id="ancestor-first">
    <title>First major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
  </topic>
  <!-- more topics in ancestor composite doc -->
  <topic id="ancestor-last">
    <title>Last major topic in ancestor composite doc</title>
    <!-- ...topic content... -->
    <topic id="ancestor-last-child">
      <title>Child of last major topic in ancestor composite doc</title>
      <!-- ...topic content... -->
    </topic>
  </topic>
</dita>

middle.dita:
<topic id="middle-root">
  <title>Root topic in middle doc</title>
  <body><!-- ... --></body>
  <topic id="middle-child">
    <title>Child of root topic in middle doc</title>
    <!-- body content -->
  </topic>
</topic>

child.dita:
<topic id="child">
  <title>Small child topic</title>
  <!-- small child topic content -->
</topic>
```

Figure 17: Source documents with nested structures

When `chunk="split"` is evaluated, both `ancestor.dita` and `middle.dita` are split and treated as multiple DITA topic documents. `child.dita` is only a single topic and has nothing to split.

- `ancestor.dita` has a root `<dita>` element, so it results in multiple peer topic references (or branches) in the map. Topic references nested within the original reference to `ancestor.dita` are now located within the reference to "ancestor-last" (the last topic child of the `<dita>` element).
- `middle.dita` has nested topics, so results in its own new hierarchy within the map. Content from the nested topic reference is now located within the reference to the root topic from `middle.dita`, but after any references to child topics.

```
input.ditamap:
<map chunk="split">
```

```
  <title>Generation example</title>
  <topicref href="ancestor-first.dita"/>
  <!-- more topics in ancestor composite doc -->
  <topicref href="ancestor-last.dita">
    <topicref href="ancestor-last-child.dita"/>
    <!-- middle.dita now located here, as final child of
         final topic child of <dita> in ancestor.dita -->
    <topicref href="middle-root.dita">
      <topicref href="middle-child.dita"/>
      <!-- child.dita now located here, as final topic
           child root topic in middle.dita ancestor.dita -->
      <topicref href="child.dita"/>
    </topicref>
  </topicref>
</map>
```

Figure 18: Equivalent source content

Example: When `@chunk` is ignored

The `@chunk` attribute is ignored in some cases, such as when `chunk="combine"` is already in effect or when `chunk="split"` is specified on a grouping element.

Ignoring `@chunk` when already combining topics

In the following example, evaluating `@chunk` results in one rendered document for each branche of the map. Any additional `@chunk` values within that branch are ignored (including `@chunk` values within any referenced maps).

```
<map>
  <title>Ignoring chunking when already combined</title>

  <topicref href="branchOne.dita" chunk="combine">
    <!-- @chunk ignored for branchOneChild.dita -->
    <topicref href="branchOneChild.dita" chunk="split"/>
  </topicref>

  <topicref href="branchTwo.dita" chunk="combine">
    <!-- Any @chunk within submap.ditamap is ignored -->
    <topicref href="submap.ditamap" format="ditamap"/>
  </topicref>
```

Figure 19: Chunk within a combined branch

Ignoring `@chunk` on a grouping element

In the following example, `chunk="split"` is specified on two grouping elements.

```
<map>
  <title>Trying to "split" groups</title>
  <topicgroup chunk="split">
    <topicref href="ingroup1.dita">...</topicref>
    <topicref href="ingroup2.dita">...</topicref>
  </topicgroup>
  <topichead chunk="split">
    <topicmeta><navtitle>Heading for a branch</navtitle></topicmeta>
    <topicref href="inhead1.dita">...</topicref>
    <topicref href="inhead2.dita">...</topicref>
  </topichead>
</map>
```

Figure 20: Chunk within a combined branch

- The `@chunk` attribute on the `<topicgroup>` is ignored; it does not cascade, and there is no referenced topic, so it has no effect.
- In some cases, an implementation might treat the `<topichead>` element as equivalent to a single title-only topic, while in other cases it might be ignored. In either case the `@chunk` value has no effect. If the `<topichead>` is treated as a title-only topic, it cannot be split further; if it is ignored for the current processing context, it is no different than the `<topicgroup>`.

Example: Combining topics within a split context

While `@chunk` attributes are ignored when a "combine" action is already in effect, it is possible to use `chunk="combine"` when `split` is otherwise in effect.

Assume the following map, where `chunk="split"` on the root element means that all topic documents within this map structure are split by default, but a branch within the map sets `chunk="combine"`.

```
<map chunk="split">
  <title>Split most, but not one branch</title>
  <topicref href="splitme.dita">...</topicref>
  <topicref href="exception.dita" chunk="combine">...</topicref>
  <topicref href="splitmetoo.dita">...</topicref>
</topicref>
```

Figure 21: Map with default "split" action, that also uses "combine"

Assume as well that no other `@chunk` attributes are specified in this map. The following points are true when `@chunk` is evaluated:

1. The document `splitme.dita` is treated as multiple split documents when it contains more than one topic. The same is true for any other document within that branch.
2. The second branch (beginning with `exception.dita`) is treated as a single DITA document, combining all topic documents within that branch.
3. The document `splitmetoo.dita` is treated as multiple split documents when it contains more than one topic. The same is true for any other document within that branch.

Example: Managing links when chunking

link management with `@chunk` is often straightforward; in most cases where it is not, using indirect links and `@keyref` will avoid ambiguity.

Input topics for following examples

The following map and topics are used for all examples in this topic.

```
<map>
  <title>Map with chunks and links</title>

  <keydef href="splitThis.dita" keys="splitThisKey"/>
  <keydef href="splitThis.dita#splitThisChild" keys="splitThisChildKey"/>

  <topicref href="splitThis.dita" chunk="split" keys="explicitSplitKey"/>
  <topicref href="combineThis.dita" keys="combineThisKey">
    <topicref href="combinedChild.dita" keys="combinedChildKey"/>
  </topicref>
</map>
```

Figure 22: input.ditamap

```
splitThis.dita:
<topic id="splitThisRoot">
  <title>Root topic in split document</title>
  <!-- ... -->
  <topic id="splitThisChild">
    <title>Child topic in split document</title>
```

```
    <!-- ... -->
  </topic>
</topic>

combineThis.dita:
<topic id="combineThisRoot">
  <title>Root topic in combined document</title>
  <!-- ... -->
  <topic id="combineThisChild">
    <title>Child topic in combined document</title>
    <!-- ... -->
  </topic>
</topic>

combinedChild.dita:
<topic id="combinedChildRoot">
  <title>Topic in child document, combined with parent</title>
  <!-- ... -->
</topic>
```

Figure 23: Topics used by input.ditamap

Topics that are rendered only once when publishing

Assume that the map above is a root map or is used by another map does not otherwise render the three topic documents. In that case, the following is true:

- `splitThis.dita` is rendered as two documents. For this example, assume a processor creates two documents with names taken from the topic ID, so that topic becomes `splitThisRoot.dita` and `splitThisChild.dita`.
- The branch with `combineThis.dita` is rendered as one document together with the content of `combinedChild.dita`. For this example, assume a processor merges the child topic into the file `combineThis.dita`.
- All links using `href="splitThis.dita"`, `keyref="splitThisKey"`, or `keyref="explicitSplitKey"` will resolve to `splitThisRoot.dita` (the only rendered instance of that topic).
- All links using `href="splitThis.dita#splitThisChild"` or `keyref="splitThisChildKey"` will resolve to `splitThisChild.dita` (the only rendered instance of that topic).
- All links using `href="combinedChild.dita"` or `keyref="combinedChildKey"` will resolve to that topic within `combineThis.dita` (the only rendered instance of that topic).

Topics that are rendered twice when publishing

Now assume that the map above is pulled into another context that also renders all three topic documents as originally authored. In that case, the following is true:

- `splitThis.dita` is rendered *in this context* as two documents. For this example, assume a processor creates two documents with names taken from the topic ID, so that topic becomes `splitThisRoot.dita` and `splitThisChild.dita`. At the same time, `splitThis.dita` is rendered *in another context* as a single document.
- The branch with `combineThis.dita` is rendered *in this context* as one document with the content of `combinedChild.dita`. At the same time, those two documents are rendered *in another context* as individual documents. For this example, assume a processor generates a combined document for this context, but a file name such as `combinedChild-2.dita` must be generated to avoid collisions with the other copy.
- All links in this map using `href="splitThis.dita"`, `href="splitThis.dita#splitThisChild"`, `href="combineThis.dita"`, or `href="combinedChild.dita"` are now ambiguous. Implementations will have to guess which topic to target, the split/combined instances from this map or the instances rendered elsewhere in the root map.

- All links using `keyref="splitThisKey"` or `keyref="splitThisChildKey"` are also ambiguous, because the key definitions are not associated explicitly with the chunked or not-chunked instance. If key scopes are used, applications might more reliably guess that the intended target is the split copy in this map, but this is not guaranteed.
- All links using `keyref="explicitSplitKey"`, `keyref="combinedThisKey"`, or `keyref="combinedChildKey"` are **unambiguous**; they can only resolve to the chunked instance from this submap.
- There is no way to unambiguously link to the child document that will result from splitting `splitThis.dita`. For details, see the edge case in Impact of chunk on linking on page 4.