White Paper: Release Management Domain

Contents

Introduction

Organizations that publish large documents, if these documents must undergo significant revision, are faced with difficult choices:

- give the reader nothing
- provide an automatically compiled, and usually unsatisfactory, list of changes
- manually maintain a revision history to inform the reader of significant changes

Usually, giving readers new revisions of large documents without some guide to what has been changed is considered unacceptable. If such documents are given to a captive (i.e., internal) audience, the result may be mere grumbling; if paying customers are involved, more serious consequences may result.

The key word in this discussion is *significant*. Machines are notoriously bad at recognizing the significance of human language, so it is nearly impossible for a computer to distinguish between trivial changes and significant ones. For large books, automatic lists or difference documents risk obscuring the significant changes in a snowstorm of trivial ones.

Up to now, manually creating revision history documents meant recording changes in external documents such as databases or spreadsheets. The DITA 1.3 release management domain facilitates the compilation of such revision histories. Without prescribing any processing, it provides elements that can be used to record release notes. The user is free to compile the actual revision history in any convenient way, such as XQuery or script, and can assemble the notes into any convenient format, such as spreadsheet, text file, or DITA topic.

Having changes recorded in topic eliminates the need for such external files.

This document will describe the release management domain.

Use of dates

An additional requirement is that release notes be filterable. In large content sets, content is frequently shared, thus increasing the likelihood that some changes apply selectively (to some output documents but not others).

The use of select attributes, while required, is insufficient by itself in some organizations; release notes may belong in one and only one document. In other words, once the release note has appeared in a release document, it should never appear in another one. Note, though, that this model is not imposed by the release management domain, which can also easily support cumulative documents.

Although the date elements are text content and no specific format is required, it is strongly recommended that the date strings used conform to the ISO 8601 standard unless a machine timestamp (for example a Unix-style timestamp) is used. The string may contain a date and time or just a date.

Example ISO 8601 date formats:

- 2013-03-06
- 2013-03-06T13:30:22.25
- 2013-03-06T13:30:22.25-05:00

Release management domain elements

Here are the elements of the release management domain. <change-historylist> is a child of prolog. It can be included in maps as a child of <metadata>.

```
change-historylist?
  change-item*
    ( change-person | change-organization )*
    change-revisionid?
    change-request-reference?
      change-request-system?
      change-request-id?
    change-started?
    change-completed?
    change-summary?
    data*
```

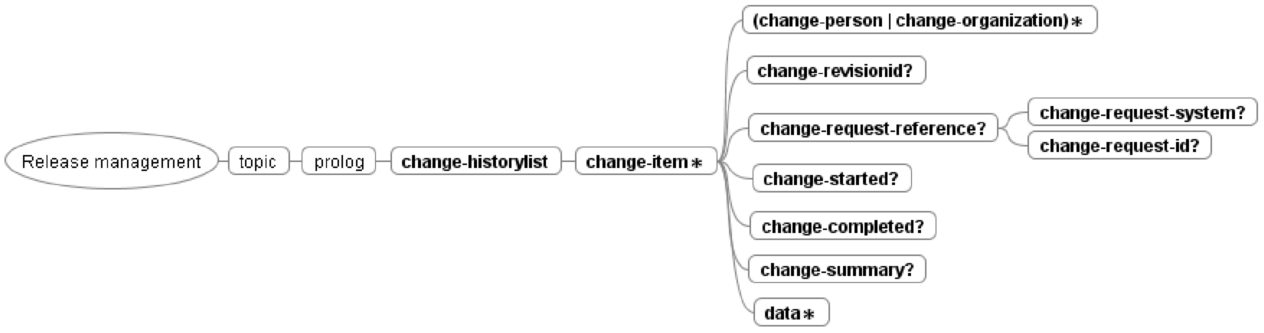

Figure 1: Release Management Elements

[[show both?]]

All these elements are derived from <data>; thus, except for the containers change-historylist, change-item, and change-request-reference, they have CDATA content models. All the elements are optional; the user is free to use as little or as much of the domain as is needed. Additional data elements of any number may be used as is or specialized to meet requirements not foreseen. All release management elements support select attribution.

Elements in detail

This section describes the elements in greater detail.

change-item	Contains a single release note. It holds information about when and by whom the topic was edited during its history.
change-person	The person making the change to the document
change-organization	An organization that requires or instigates a change. Examples include government agencies or standards bodies.
change-revisionid?	Contains an identifier associated with the change described by the release note

change-request-reference?	Changes may result from tickets filed in defect tracking systems or other databases. This element is a container for the next two elements
change-request-system?	The tracking system or database from which the change originated (see change-request-reference)
change-request-id?	The id or other key number linking the change back to the tracking system or database (see change-request-reference)
change-started?	The date work on the change began. Recommended date format is ISO-8601, with or without time information, (for example 2014-06-17) unless a machine timestamp is used.
change-completed?	The date work on the change was completed. Recommended date format is ISO-8601, with or without time information, (for example 2014-06-17) unless a machine timestamp is used.
change-summary?	A text description of the change. This should represent the actual text describing the change as presented to the reader.

Examples

Example 1

This figure shows three simple release notes added to a single topic. This topic is used in documentation for two products, A and B.

```
<prolog>
...
  <changehistory-list>
    <change-item product="productA productB">
      <change-person>Bill Carter</change-person>
      <change-completed>2013-03-23</change-completed>
      <change-summary>Made change 1 to both products</change-summary>
      <data>Details of change 1</data>
    </change-item>
    <change-item product="productA">
      <change-person>Bill Carter</change-person>
      <change-completed>2013-06-07</change-completed>
      <change-summary>Made change 2 to product A</change-summary>
      <data>Details of change 2</data>
    </change-item>
    <change-item product="productA productB">
      <change-person>Bill Carter</change-person>
      <change-completed>2013-07-20</change-completed>
      <change-summary>Made change 3 to both products</change-summary>
      <data>Details of change 3</data>
    </change-item>
  </changehistory-list>
...
</prolog>
```

Figure 2: Excerpt from prolog of topic myTopic

Examples showing date usage

Example 2

To illustrate the use of date filtering, in this scenario revision 5 of product A's manual was published on June 1, while product B's manual hasn't been published since February 10 (revision 2). Then, on September 3, both manuals are being published. Here is a timeline of events:

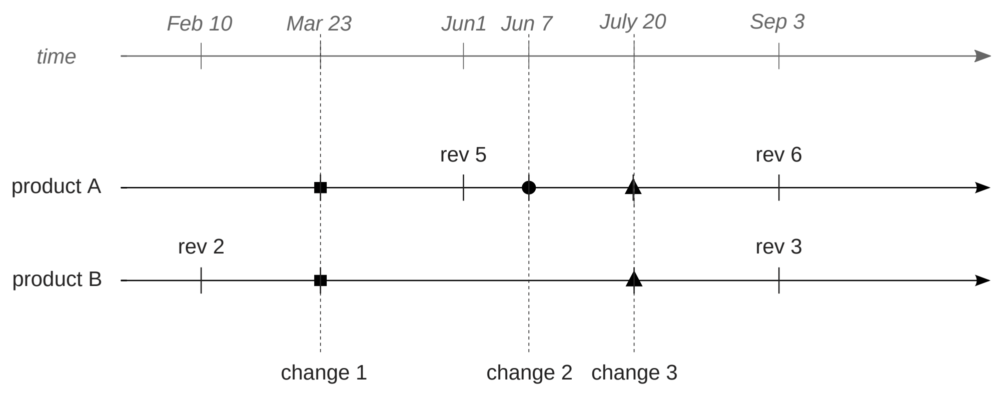

Figure 3: Example 2 timeline

Thus, product A's release notes for revision 6 should include only those changes since June 2, while those for revision 2 of product B should start with changes made on February 11. Here is what these documents' release notes should contain for this topic:

Table 1: Excerpt from product A's revision 6 release notes, September 3 (last published June 1)

Topic changed	**details**
myTopic	Made change 2 to product A
myTopic	Made change 3 to both products

Table 2: Excerpt from product B's revision 3 release notes, September 3 (last published February 10)

Topic changed	**details**
myTopic	Made change 1 to both products
myTopic	Made change 3 to both products

Note that change 1 already appeared in the revision 5 release notes of product A on June 1. Therefore, it must *not* appear in the revision 6 release notes, or it will alert the customer to something that hasn't actually changed since the previous revision.

Achieving this goal with select attributes alone would come at the cost of additional attribute values for not only each product, but for each revision of each product—which would become unmanageable when instead of two products there were 35, and instead of one topic 120,000.