White Paper: Release Management Domain

Contents

The problem..... 3
Managing release data..... 3
In DITA..... 3

The solution..... 4
Introducing the release management domain..... 4
Release management domain elements..... 4
Elements in detail..... 5
Example release notes..... 5
Sample output showing date filtering..... 6

The problem

Managing release data

Documenting the workings of complex machines--a jet fighter or a CAT scanner--requires the marshaling of thousands of pieces of information. Even with a very low error rate, corrections will be required. In a large document, a revision could mean hundreds of changes. Most readers will complain if they are given an document update without a list of its significant changes.

The key word here 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.

Recording these changes then becomes the responsibility of the content worker.

In DITA

As the use of DITA spreads into more industries and complex documents, a bit of help for its practitioners is welcome. Since the release of DITA 1.0, those who use DITA for new releases of products and documents have had to resort to workarounds: spreadsheets, text files, or maybe a special-purpose topic. Some lucky few might have a content management system with good metadata facilities. But the release information always external to the content.

With the development of DITA 1.3, we have implemented a method of recording the changes in the topic itself. Release management data in the topic offers many advantages:

- For authors, it eliminates the time-consuming and error-prone step of opening a separate topic, spreadsheet, or other document and recording the changes there. If a cross reference is needed, the cross reference could be added to the change note in the DITA topic and not as a separate process.
- It removes the need to cross track change notations between topics and a record-keeping system.
- It eliminates reliance on CMS metadata to track changes.
- For readers, it means that in all likelihood, they will receive more accurate documents since compliance with release management requirements will improve.
- In a topic-based display, such as infocenter or wiki, it would facilitate a tabbed display: one tab for content, one for history. (Wikipedia has such a display.)
- It will facilitate adding detailed information about changes directly into the topic or the map
- It will allow automated production of release notes or other documents, especially those required by regulatory bodies

In this feature article, we introduce the domain, describe its elements, and give examples of its use.

Available with this document is an XQuery and some sample files. The XQuery follows a DITA map and extracts release notes from the DITA topics in the map. The release notes are placed in a table in a newly generated topic that can be included in a publication or used alone.

The solution

Introducing the release management domain

The release management domain in DITA 1.2 is based on the book change-history element in the DITA bookmap. The DITA 1.2 release management domain is included as metadata in the prolog of DITA topics and in the map metadata.

Release management is an element-only domain; current DITA Open Toolkit processing does not output the new elements. To output the data in the release management domain, you must provide your own processing. We have provided the XQuery example as one method of processing the data.

All the release management elements are optional. They all support conditional-processing attributes as well, since the domain was intended for use in shared document environments.

For some organizations, the use of conditional-processing attributes is insufficient by itself. In some organizations, release notes are kept in one and only one version of a document. In other words, once the release note has appeared in print, it should never appear in another version of the document. Note, however, that this model is not imposed by the release management domain, which can also easily support cumulative release notes.

Release management domain elements

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

[how to present?]

```
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*
```

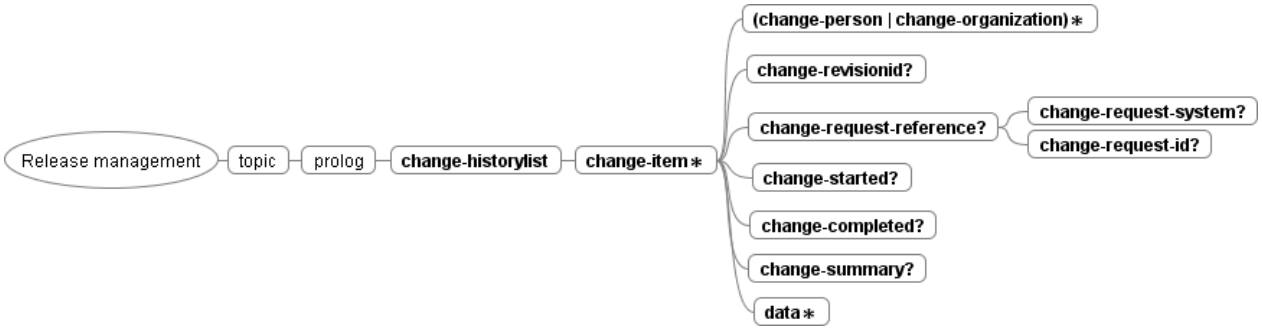

Figure 1: Release Management Elements

[[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 additional requirements. All release management elements support conditional-processing attributes..

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	Names the person making the change to the document.
change-organization	Names the organization that requires or instigates a change. Examples include company departments or regulatory bodies.
change-revision-id?	Contains an identifier associated with the change described by the release note such as an individual's secure and unique ID, a System Change Request (SCR) number, or a Hazard Mitigation Number (HM).
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?	Names tracking system or database from which the change originated (see change-request-reference).
change-request-id?	Names the id or other key number linking the change back to the tracking system or database (see change-request-reference).
change-started?	Names the date work on the change began. The recommended date format uses the format in ISO-8601, with or without time information. An ISO-compatible date appears as 2014-06-17 unless a machine timestamp is used.
change-completed?	Names the date work on the change was completed. The recommended date format uses the format in ISO-8601, with or without time information. An ISO-compatible data appears as 2014-06-17 unless a machine timestamp is used.
change-summary?	Provides a text description of the change. This description should contain the actual text used to describe the change to the reader.

Example release notes

This example shows three simple release notes added to a single topic. This topic is used in release-management information 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>Phil 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

Sample output showing date filtering

One presentation of the data from the example release notes might be as a table. The sample XQuery outputs a topic containing such a table.

Here is an illustration of 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 published. Here is a timeline of events:

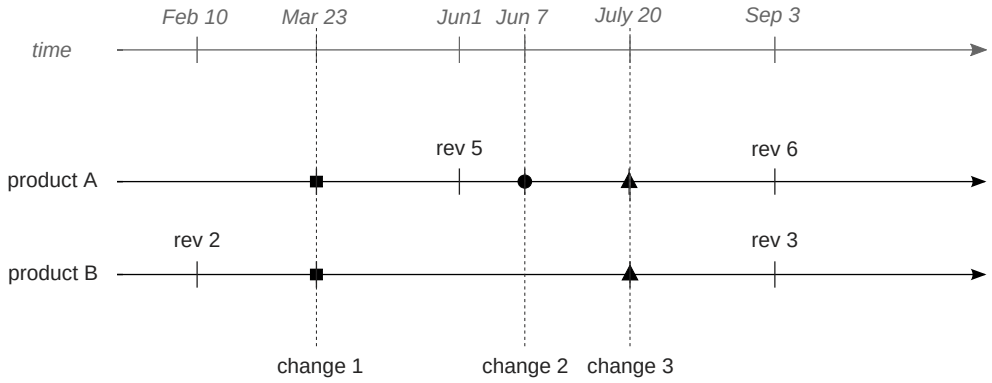

Figure 3: Example timeline

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)

Change Site	details
Topic X	Made change 2 to product A
Topic X	Made change 3 to both products

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

Change Site	details
Topic X	Made change 1 to both products
Topic X	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 should *not* appear in the revision 6 release notes,or it may mislead customers by alerting them to something that hasn't actually changed since the previous revision.