

An OASIS DITA Adoption Technical Committee Publication

DITA 1.3 Feature Article: About the DITA 1.3 release management domain

Author: Tom Cihak, Keith Schengili-Roberts
On behalf of the DITA Adoption Technical Committee

Date: 3 June 2015

This is a Non-Standards Track Work Product and is not subject to the patent provisions of the OASIS IPR Policy.



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

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

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

This white paper was produced and approved by the OASIS DITA Adoption Technical Committee as a Committee Draft. It has not been reviewed and/or approved by the OASIS membership at-large.

Copyright © 2015 OASIS. All rights reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website. This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English. The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Document History

Revision	Date	Author	Summary
First Draft	xx xxxx 2014	Cihak	Initial draft
Second Draft	05 Dec 2014	Cihak	All comments incorporated
Third Draft	27 Feb 2015	Keith S-R	Copyedit pass, more context and new example added
Fourth Draft	3 Jun 2015	Keith S-R	Typo and code example formatting fixed, conclusion added

Revision	Date	Author	Summary
Fifth Draft	20 Jul 2015	Keith S-R	Added map example based on comment from Eliot Kimber

Table of Contents

The problem.....	5
Managing publication release data.....	5
Release Management in DITA 1.3.....	5
The solution.....	7
Introducing the release management domain.....	7
Release management domain elements.....	7
Release Management elements in detail.....	8
Release note example #1.....	9
Release note example #2.....	10
Release note example #3.....	10
Sample output showing date filtering.....	11
Conclusion.....	13

The problem

■ In this feature article, we describe the problems the release management domain is intended to solve.

Managing publication release data

Documenting the workings of complex machines—such as 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 complain if they are given a 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 those that human readers would consider to be significant. For large technical publications, simply providing automated lists of differences risks obscuring any significant changes amidst a myriad of trivial ones.

Recording and keeping track of these changes is the responsibility of those producing technical content. This process is often done manually, and it is often prone to error, as content creators may forget to add significant additions, misremember which update was added when, or simply not know what has been added by others in those cases where many people have worked on a single documentation project.

Having a convenient process for automatically and reliably handling release-specific information was a goal in DITA 1.3, ideally one allowing individual content authors to note significant changes at the topic or map level and then create output automatically based on the parameters provided.

Release Management in DITA 1.3

As the use of DITA spreads to more industries with their own complex document requirements, additional help for efficient release management for its practitioners is welcome. Up until now those who use DITA for handling the new releases of products and documents have had to resort to workarounds: spreadsheets, text files, or special-purpose topics. A lucky few may have access to a content management system with good metadata facilities. But the release information has always been external to the content.

With the development of the release management domain in DITA 1.3, there is now a method of recording significant changes within topics and maps.

Release management data in topics and maps offers many advantages:

- For content authors, it eliminates the time-consuming and error-prone step of opening a separate topic, spreadsheet, or other document and recording significant per-topic changes there.
- Cross references can be added to the change note within the DITA topic and not as a separate process.
- Provides a consistent and robust method for content authors to describe significant changes within a topic or map.
- Reduces the need for CMS metadata to track changes.
- Readers receive more accurate descriptions of document changes.
- For an infocenter or wiki, it can facilitate a tabbed display: one tab for content, one for history. (Wikipedia has such a display.)
- It enables the automated production of release notes or other related documents, especially those required by regulatory bodies for documentation compliance.

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 within the target 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

This section describes how the release management domain seeks to solve the problem.

Introducing the release management domain

The release management domain in DITA 1.3 is based on the book change-history element in the DITA bookmap that was first introduced with DITA 1.2. The release management elements can be used the prolog of DITA topics and in the bookmeta of a bookmap.

Release management is an element 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 of 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 many cases, release notes are kept in only one version of a document. In other words, once the release note has appeared in print, its contents do not appear in subsequent versions of the document. Note that this model is not imposed by the release management domain, which can easily support cumulative release notes.

Release management domain elements

The <change-historylist> element is a child of <prolog>. It can also be included in bookmaps as a child of the <bookmeta> element.

The following XML trees depict the relationships of each element. Elements that take a single value are denoted by a question mark symbol, and those with multiple possible values with an asterisk.

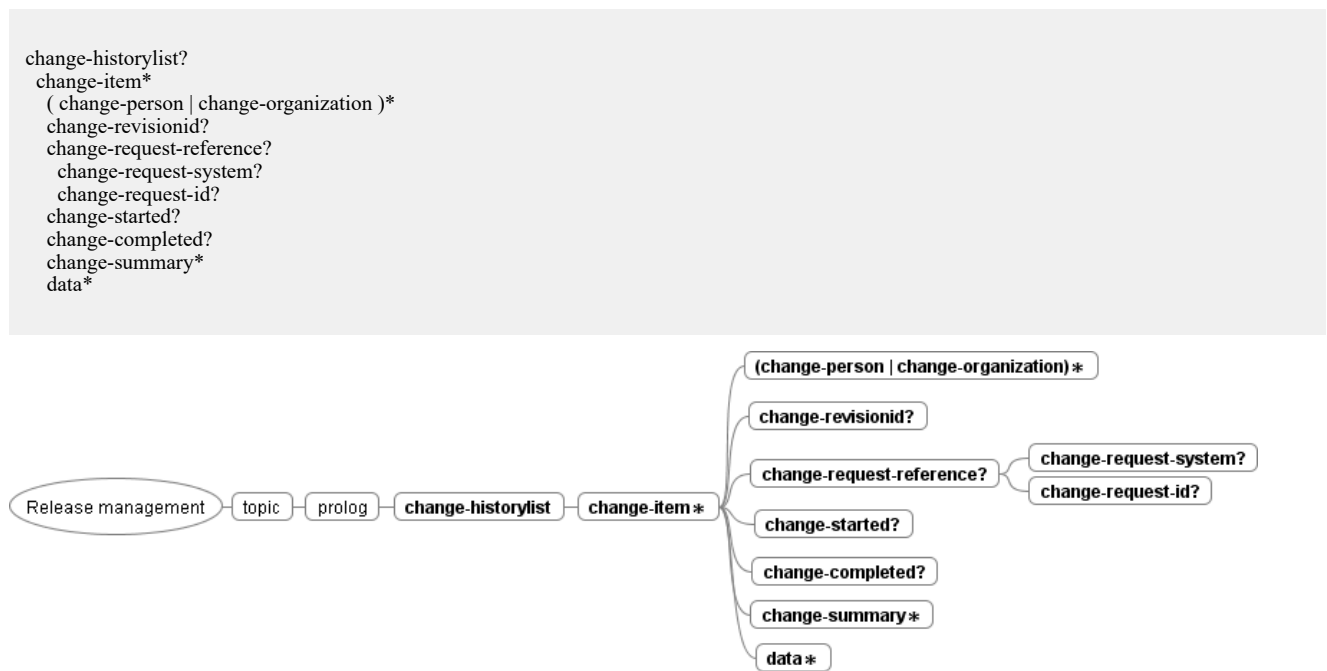


Figure 1: Release Management Elements

All these elements are derived from the <data> element. Thus, except for the containers <change-historylist>, <change-item>, and <change-request-reference>, they have CDATA content models. Because the elements are all optional, the user is free to use as little or as much of the domain as is needed. Additional data elements may be used as is or specialized to meet any additional requirements. All release management elements support conditional-processing attributes.

Release Management elements in detail

All release management elements are optional, and in the following list elements with single values are denoted by a question mark symbol, and multiple possible values with an asterisk.



Note: The <data> element is omitted as it is not specific to the release management domain.

<change-item>*

Contains a single release note. It holds information about when and by whom the topic was edited during its history. This element (and all of the others in this list) can also take on all of the standard metadata attributes, such as @product, @audience, @platform, etc.

<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, a Hazard Mitigation Number (HM), or any other user-defined revision ID.

<change-request-reference>?

Significant changes may result from bug tracking tickets filed in defect tracking systems or related databases. This element is a container for the next two elements.

<change-request-system>?

Names the 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 <changerequest-reference>).

<change-started>?

Names the date work on the change began. The recommended date format uses the ISO-8601 format, with or without time information. An ISO-compatible date for June 17, 2014 would appear as “2014-06-17” unless a machine-generated timestamp is used instead.

<change-completed>?

Names the date work on the change was completed. The recommended date format uses the ISO-8601 format, with or without time information. An ISO-compatible data for June 16, 2017 would appear as “2014-06-17” unless a machine-generated timestamp is used instead.

<change-summary>*

Provides a text description of the change. This description should contain the text used to describe the change to the reader.

Release note example #1

The following simplified example shows two release notes added to a single topic for a single unnamed product. It provides an illustration of how all of the elements within the release management domain might be used.

```
<prolog>
...
<changehistory-list>
  <change-item>
    <change-person>John Smythe</change-person>
    <change-organization>Engineering</change-organization>
    <change-revisionid>topic-change-001</change-revisionid>
    <change-request-reference>
      <change-request-system>BugTracker Pro</change-request-system>
      <change-request-id>BT001</change-request-id>
    </change-request-reference>
    <change-started>2014-10-15T16:03:17-05:00</change-started>
    <change-completed>2014-10-22T10:51:52-05:00</change-completed>
    <change-summary>Description of new foo feature added.</change-summary>
    <data>New feature addition for v3, originally relating from a UI change request that came in from a customer</data>
  </change-item>
  <change-item>
    <change-person>John Smythe</change-person>
    <change-organization>Engineering</change-organization>
    <change-revisionid>topic-change-002</change-revisionid>
    <change-request-reference>
      <change-request-system>BugTracker Pro</change-request-system>
      <change-request-id>BT002</change-request-id>
    </change-request-reference>
    <change-started>2014-10-16T03:17-05:00</change-started>
    <change-completed>2015-02-17T15:12:41-05:00</change-completed>
    <change-summary>Description of new foobar feature added.</change-summary>
    <data>New feature addition for v2, originally relating from feature request that came in from a customer</data>
  </change-item>
</changehistory-list>
...
</prolog>
```

Figure 2: Excerpt from the prolog of a topic that uses all of the release management elements

In this case the topic contains two separate change items, which describe all of the following:

- The author of both of the changes to the documentation: "John Smythe".
- The organization from which the change came: "Engineering".
- The revision ID for each change to the topic: "topic-change-001" and "topic-change-002".
- The name of the system where the change request originated: "BugTracker Pro".
- The change request ID from that system: "BT001" and "BT002".
- Time that work on the change was started, in ISO-8601 date/time format including a time stamp offset from UTC by -5 hours (EST/EDT time zone).
- Time that work on the change was completed, using the same in ISO-8601 date/time format).
- Text description of the significant change to the topic. This content is aimed at the reader of the final published release note.
- Additional descriptive data meant for internal purposes only.

When processed for output, this topic could display the contents of its multiple change-summary elements, which could be presented in a bulleted list, as in the following example:

[title of topic]

- Description of new foo feature added.
- Description of new foobar feature added.

Release note example #2

This example shows three simple release notes added to a single topic. This topic is included 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>Phil Ford</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 3: Excerpt from prolog of topic "myTopic"

Release note example #3

The following example shows two release notes added to a map. Within a topic, release management content is contained within <prolog>. In a bookmap, release management material is inserted within <bookmeta>. Within a map, use <topicmeta>.

In the following example, the release note content tracks changes to the name of the publication over time.

```
<bookmeta>
...
<changehistory-list>
  <change-item>
    <change-completed>2015-07-20T11:51:14-05:00</change-completed>
    <change-summary>Document title updated from "Widget Repair" to "Widget X2 Repair".</change-summary>
    <data>Product has been updated and the document has been expanded with fresh examples of widget repair.</data>
  </change-item>
  <change-item>
    <change-completed>2013-11-01T18:32:52-05:00</change-completed>
    <change-summary>Original document title changed from "Foobar Repair" to "Widget Repair"</change-summary>
    <data>Original product name has been changed, with "Widget" the official product name. Content has also been updated to include the latest content up to the time of its official release.</data>
  </change-item>
```

```
</changehistory-list>
...
</bookmeta>
```

Figure 4: Excerpt from the bookmeta of the map "myMap"

Sample output showing date filtering

This topic contains more examples of release notes using the release management domain.

One presentation of the data from a release note 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:

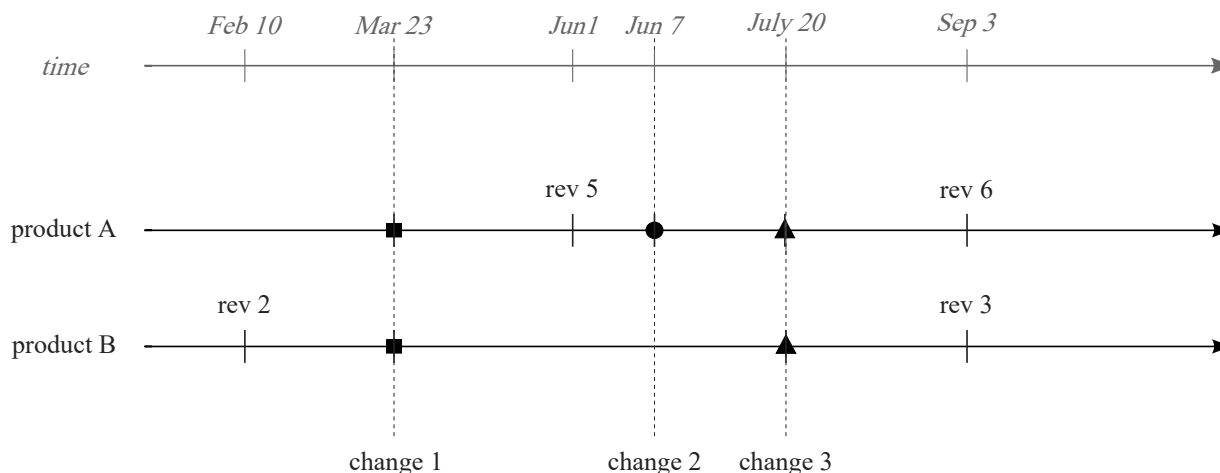


Figure 5: Example timeline

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.

Conclusion

The new release management domain in DITA 1.3 should prove to be a welcome addition for technical documentation departments seeking a solution for the problem of producing accurate release note content. As content is intended to be added at the topic and map level as it is being written, technical writers no longer have to rely on memory alone to document the major changes of a release. The date and time information embedded in the release management content allows for scripts to programmatically select only those comments that are pertinent to a given release, while still keeping information about all successive significant changes over the development of a product. Additional, optional information relating to bug tracking systems, the change requesters, and related information allows for tie-ins to external systems, and for authors to easily determine the source of significant changes. In short, the release management domain provides a convenient process for automatically and reliably handling release-specific information.