5ˇº˚Şˇ#Đ ˚ı˚Æˇæ ˇı×#ı¸# ,7$#

A WHITE PAPER BY ASTORIA SOFTWARE

Managing multiple releases when using DITA can be quite a challenge. The usual technique is to branch a document for a specific release and then merge the branch with the trunk to incorporate changes introduced in the branch. This approach works for large, monolithic documents with little or no reuse. However, DITA emphasis in content reuse imposes substantial processing requirements on branch/merge. Branching a DITA-based book means creating a completely new set of maps and topics, and all content linkages must be reworked to use the new set of assets. Merging duplicates the re-linking effort and adds conflict recognition and resolution. For these reasons, commercial content management systems merge at most two branches in a single operation. As the number of branches increases, the effort to merge possibly thousands of branched files quickly overtaxes the cumbersome branch/merge approach.

Astoria Release Labels are an elegant method of release management for DITA content that resolves the problems of the branch/merge approach. Release Labels are applied to element-level versions, providing the ability to simultaneously create and update *any number* of versions.

Contents

The Challenge of Release Management with DITA

Managing multiple releases when using DITA can be quite a challenge. Traditionally, branch/merge was the only release management method available for data models that used large monolithic documents with little or no reuse. The adoption of DITA, with its hundreds or thousands of individual XML files all connected through various link methods, has exposed a weakness in the branch/merge approach. To truly branch a DITA-based book, a completely new set of maps and topics would have to be created, and all links would have to be reworked to use the new set of assets. Furthermore, branch/merge forces decisions on handling common and collection files (i.e., should they be branched or not), and it must be done in such a way to enable a future merge (which would again rework all the links, etc.) and still provide the kind of automated conflict recognition and notification that a component content management system provides.

Additionally, Commercial branch/merge implementations can only merge two branches in a single operation. If there are more than two branches to be merged, then the merge operation must be broken into sub-steps that merge just two branches at a time. For example, a five-way documentation branch would require four merge operations in sequence to produce a newly consolidated baseline. In practice, the effort to merge possibly thousands of branched files quickly overtaxes the cumbersome branch/merge approach.

Astoria provides an elegant method of release management that resolves these two problems with branch/merge-based release management. The Astoria solution does not require creating new data sets. Instead, Release Labels are applied to element-level versions, providing the ability to simultaneously create and update *any number* of versions.

Astoria Release Labels

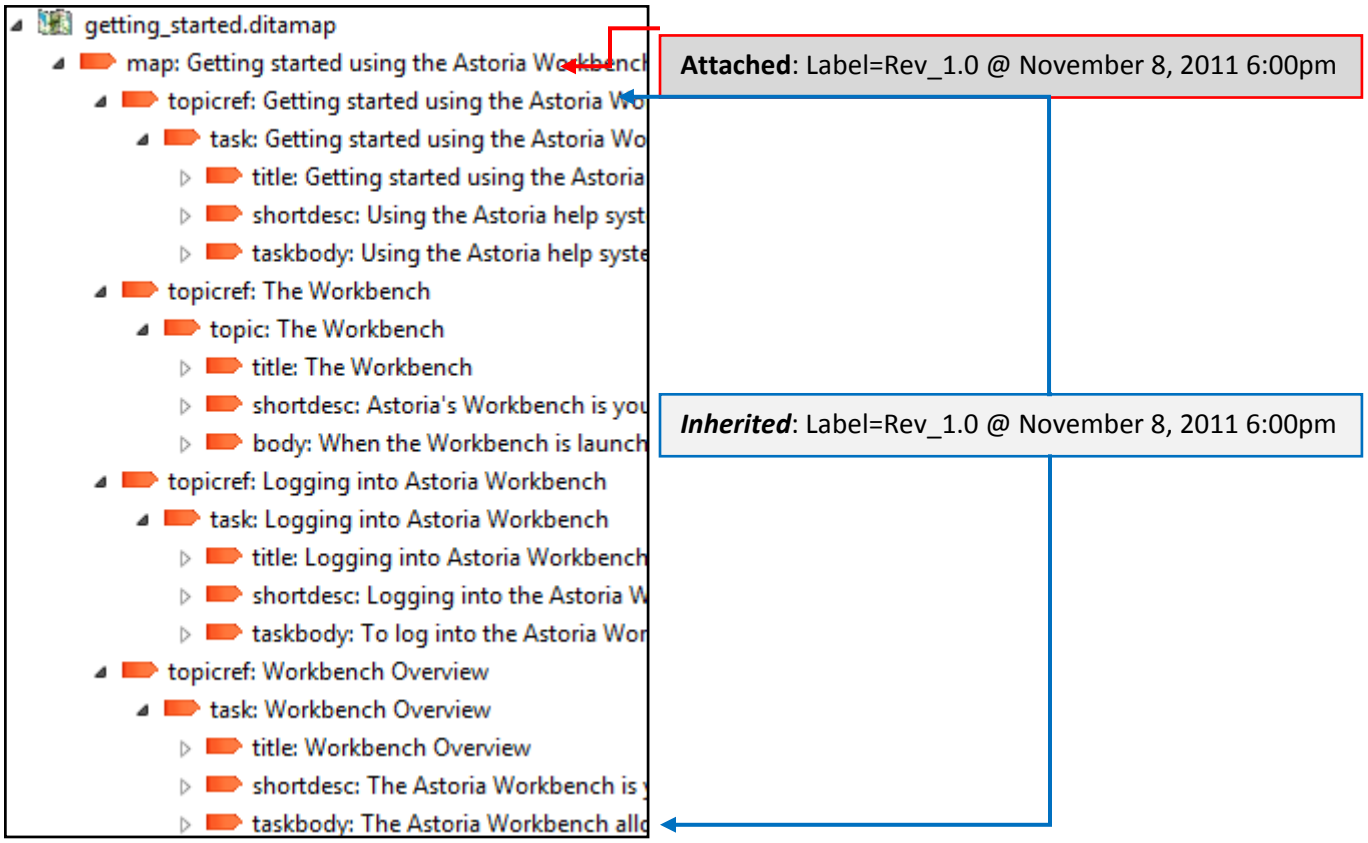
Release Labels are Astoria metadata that can be attached to any element in the Astoria repository. A release label has two attributes: a Label Name and a Timestamp. The release label is attached to any element. The Label's Timestamp is then used as a check-point for determining which versions of every descendent element is part of that release. These check-points propagate to all descendent elements, including those in referenced submaps, topics, conref'd elements and referenced graphics.

Release Labels ***end*** the practice of "locking down" a document set while publishing or translations is underway. The Release Label check-points allow continuous and simultaneous editing and authoring of content in multiple stages of development or release. In other words, users can edit files for an upcoming release while retaining the capability to publish or translate earlier releases of those same files.

Release Label Inheritance

Release Label inheritance provides the basis to determine each descendent element's version to be used when processing.

For example, suppose a Release Label with Name=**Rev_1.0** and time-stamp=November 8, 6:00pm PDT is placed on the <map> element of *getting_started.ditamap*. In Astoria, every descendent element of that map inherits the map's Release Label and time-stamp. The following image illustrates this concept:

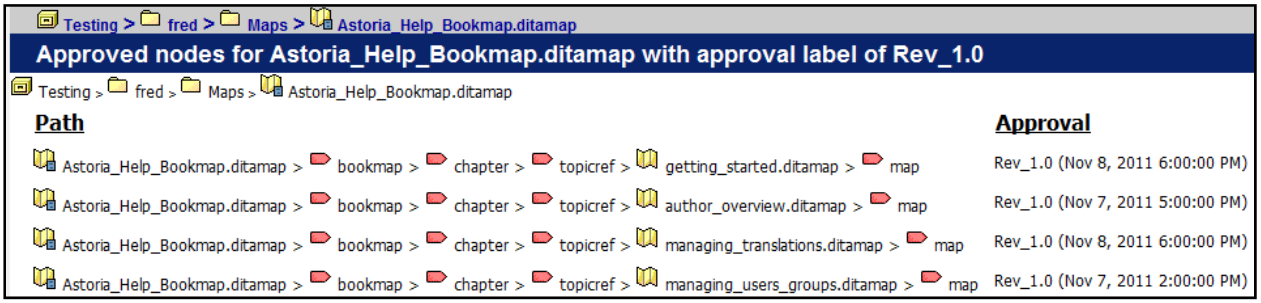


With the Release Label in place, it is possible to process only those element versions that were current at the time indicated in the time-stamp.

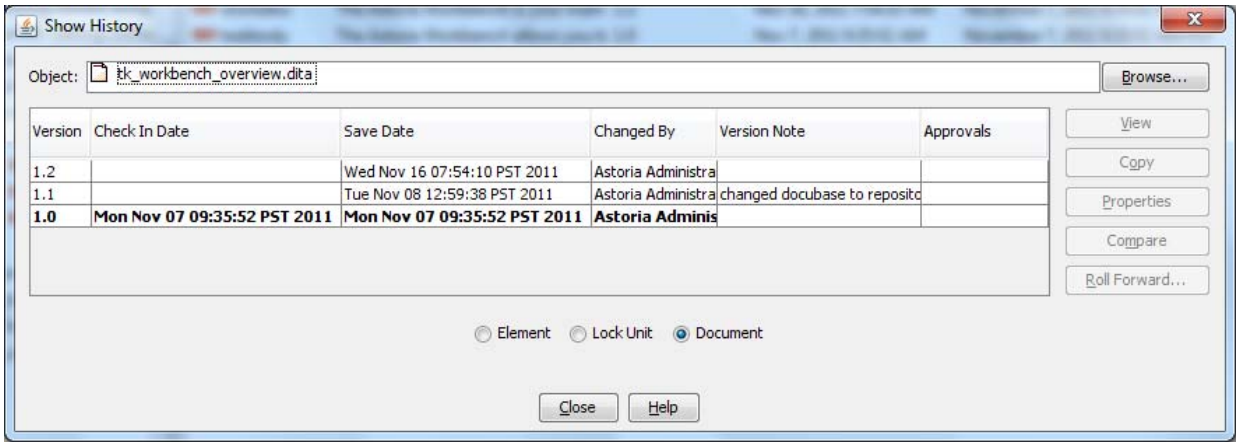
Release Labels and element versions

Files and elements are versioned based on the number of times they are checked-out, changes made, and then checked-in. Each version has a descriptive label (e.g. 1.0), a descriptive comment and a date. Between product releases the rate of change for the document set can range from never changed to many changes, where each check in results in a new version.

When a product is released and a Release Label is attached to any ancestor element, the Label's time-stamp determines which version of each element to use – the version that was current at the time indicated by the label time-stamp. Release Label inheritance is the key to making this work easily. In the aforementioned example, elements inherit their Release Label from the chapter map *getting_started.ditamap* <map> tag, where label **Rev_1.0** has a time-stamp of Nov 8, 2011 @ 6pm.



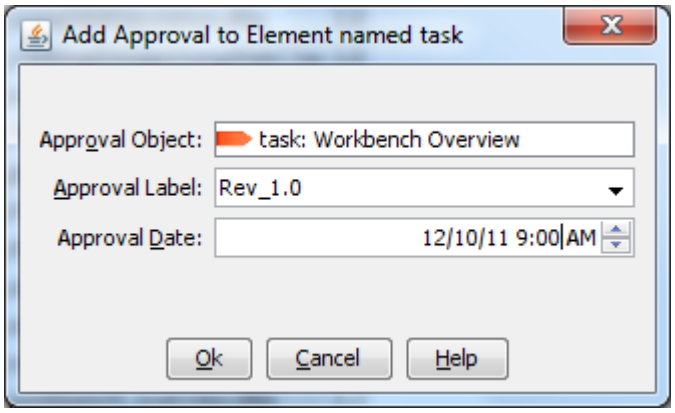
The task *tk_workbench_overview.dita* is referenced by the map *getting_started.ditamap*. Looking at this topic's history, notice that this topic has been checked-in three times. Version 1.2 was checked-in after the date associated with the Release Label **Rev_1.0** of the <map> tag, which is an ancestor of this document. If this Release Label were used to filter the data for a specific purpose, such as translation or composition, the content of version 1.1 (not the current version1.2) would be used.



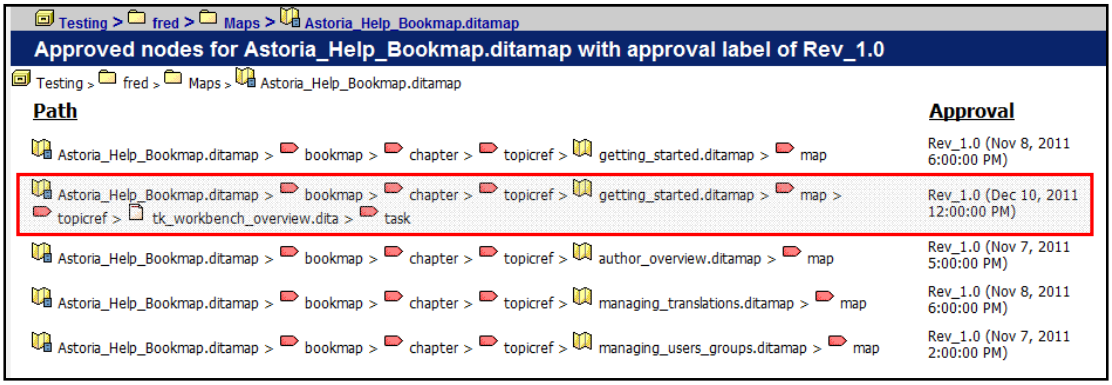
Linking the new changes to a specific release

The changes made to this file on "Nov 8" and "Nov 16" can be merged into the Rev_1.0 release by simply applying the Release Label **Rev_1.0** to the topic itself (instead of relying on an inherited Release Label).

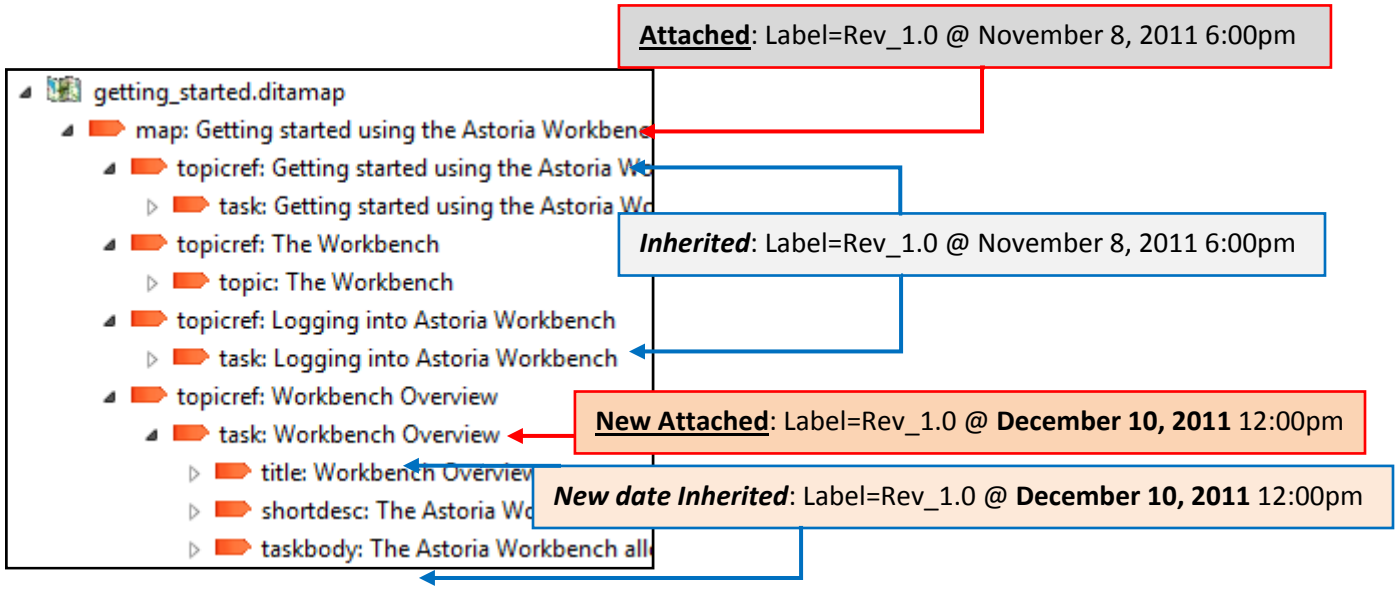
To add (merge) the changes made in the document *tk_workbench_overview.dita*, add the same Release Label **Rev_1.0** with a date that includes the current version of the document (version1.2 on November 16). The Release Label is added to the root element of the document.



The Release Report for **Rev_1.0** now shows that for this document, the "inherited" **Rev_1.0** date will be over-ridden by **the Rev_1.0** date of the <task> element. The label's date of November 8 @ 6pm on the <map> will be changed to December 10 @ 12pm starting with the <task> element, which will be applied to it and any descendent element in *tk_workbench_overview.dita* – including referenced items.



The principle is if a descendent element has a Release Label with the ***same name but a different time-stamp***, then the time-stamp of that attached Release Label will be used for that element and all its descendent elements, overriding the time-stamp that was set in the ancestor element. For example, the following map has the Release Label **Rev_1.0** set on the <map> element. The Release Label is inherited by all descendent elements ***except*** those in the fourth topic because the fourth topic has the ***same named*** Release Label with a different time-stamp. The time-stamp set for the Release Label on the <task> element is inherited by all the descendents of that <task> element.



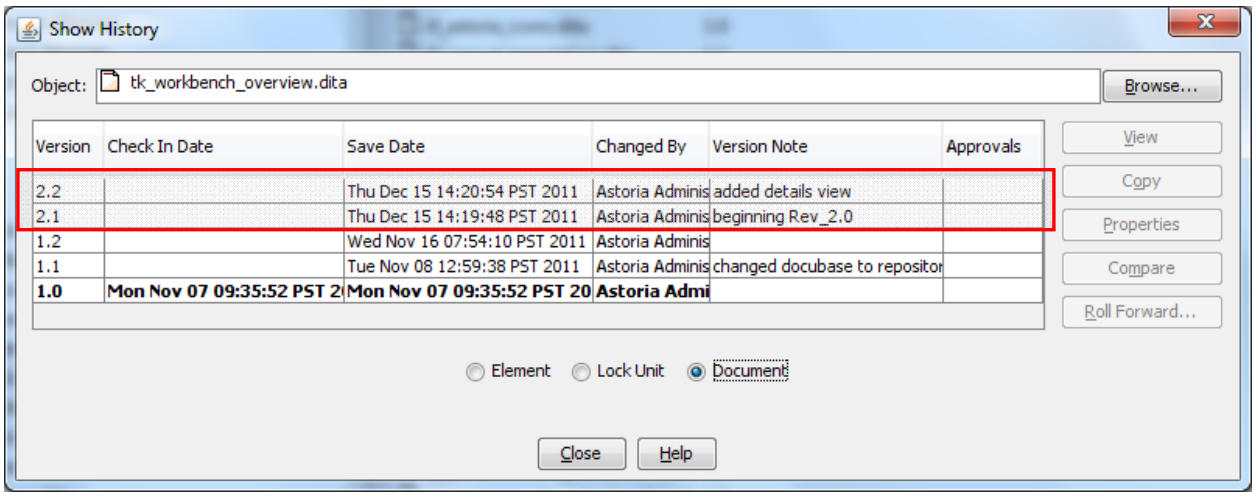
Managing Multiple Releases with a Single Document Set

Release Labels are the foundation for managing ***any number of releases*** contained in a single document set. Documents may accumulate multiple Release Labels, each with a specific label name and time-stamp. As earlier releases get updated with new information and corrections, users update the Label Names with more recent time-stamps. In publishing, translation, and export operations, users select content by simply choosing a Release Label, and the system picks the appropriate versions of all content bearing that label (either by direct attachment or by inheritance). The Release Label technique allows for easy management of updates across any of the multiple releases that users are responsible to maintain within the document set.

Multiple release management

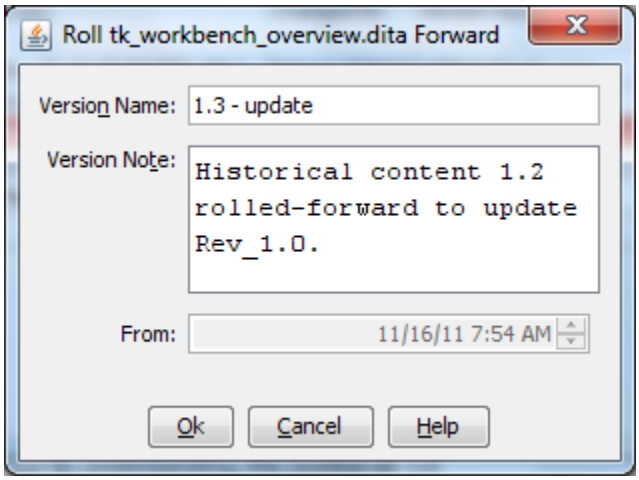
At any time during this process, authors can work on the same document set for any release. There is no need to duplicate or branch any of the documents. Release Labels provide all the appropriate check-points so that each Release constitutes a set of specific versions of each document and element in the same document set.

Consider the following example. The document *tk_workbench_overview.dita* is now changed in preparation for **Rev_2.0**. It has been checked-in twice since the additions for **Rev_1.0** were made.

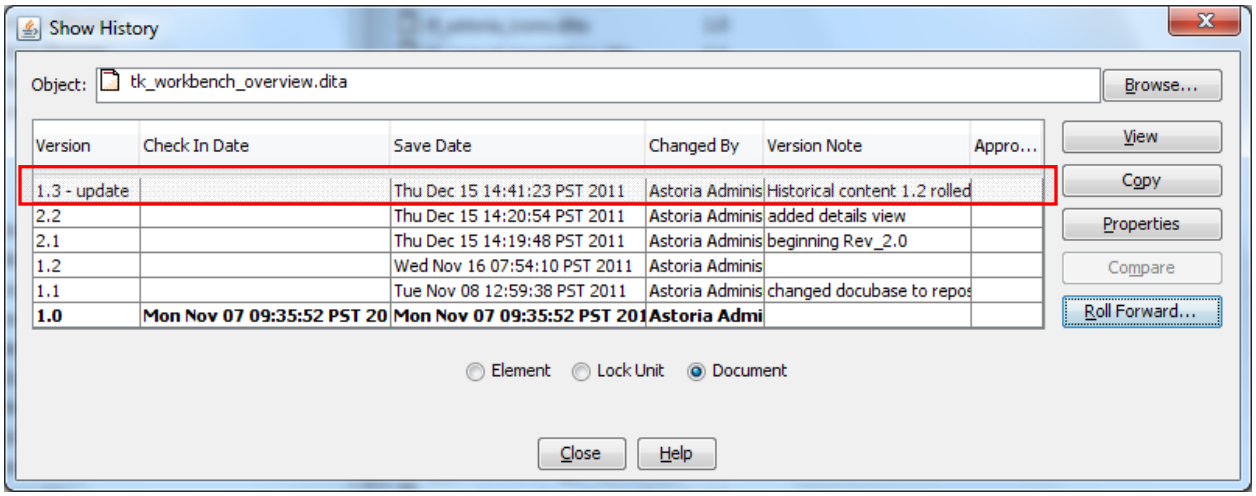


The document can still be updated for any changes for **Rev_1.0**. The document has been changed considerably for the next release **Rev_2.0**. The changes requested for **Rev_1.0** are only for **Rev_1.0** and should not be included in **Rev_2.0**.

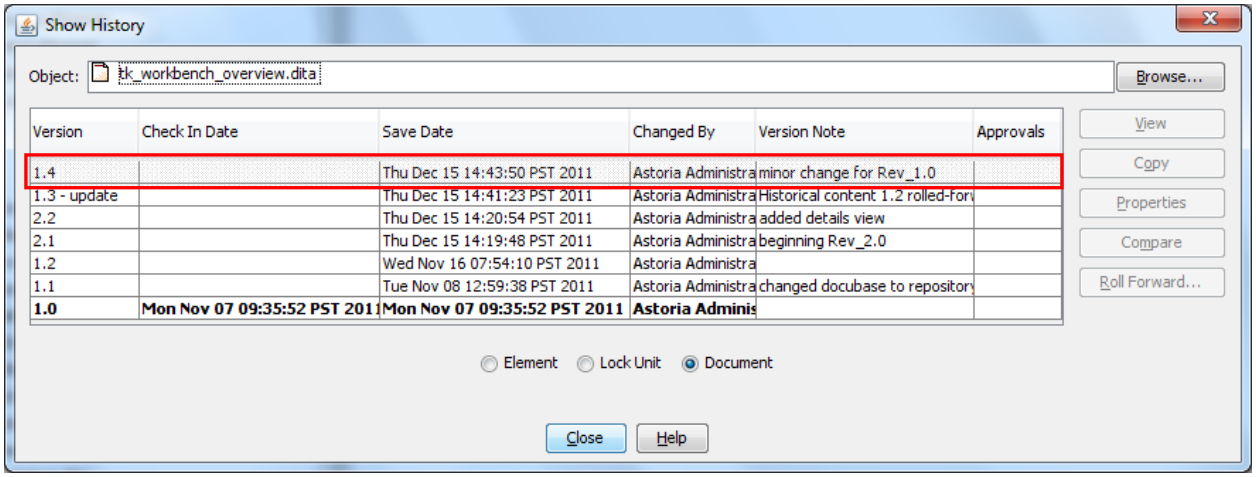
The first step is to make the last version of the document that was part of **Rev_1.0** the current version so it can be changed. This is done by rolling forward version 1.2. So that other users understand what is happening, the version label is modified to indicate this is only for the 1.0 release.



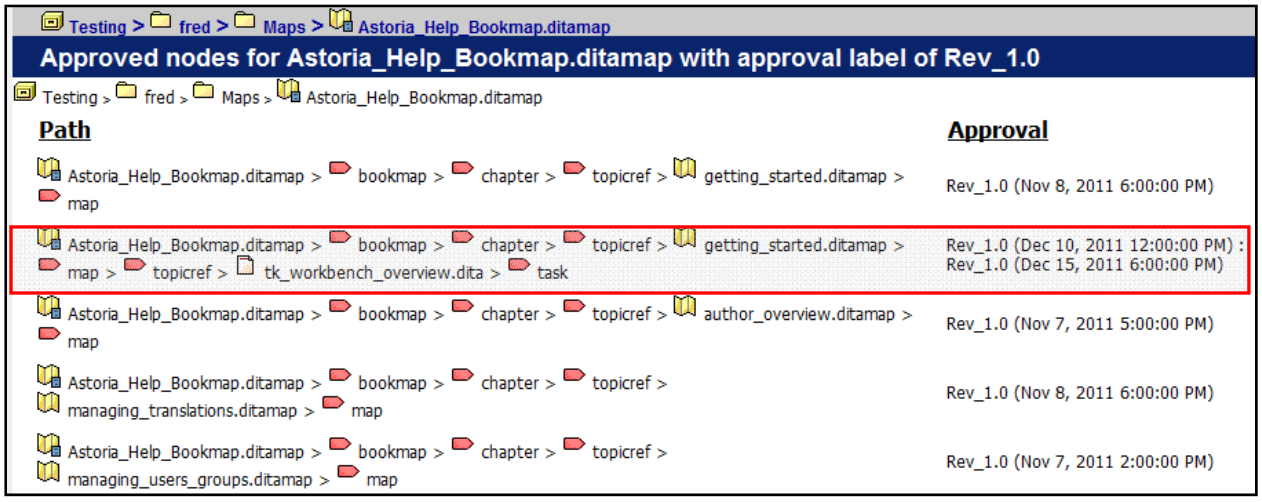
A user can now edit the last approved version for the release **Rev_1.0**.



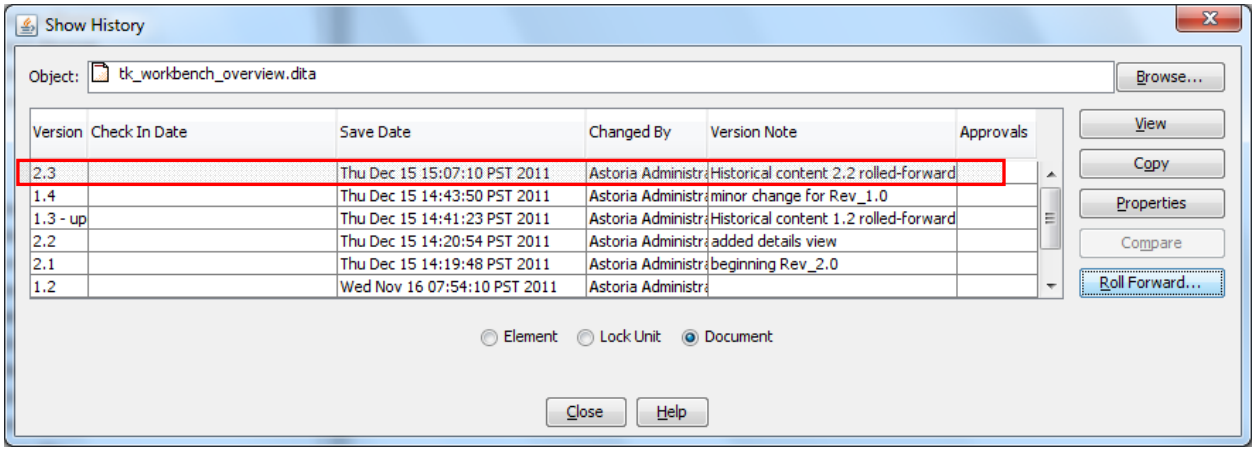
Once the file has been edited, it is checked in as a change for **Rev_1.0**.



To merge this change into **Rev_1.0**, the Release Label **Rev_1.0** with a *new* date is added to the topic. When the label **Rev_1.0** is used for processing, there will be two labels on this document with the same name. The last one attached will be the one used, in this case the December 15 time-stamp.



The changes are now part of **Rev_1.0**. The previous version of the file that includes the content for the next release, **Rev_2.0**, is now restored as the current release – leaving the document like it was before the update for **Rev_1.0** was done. Notice that version numbering has no meaning to Astoria. It is arbitrary, descriptive metadata. The order, date and author are what matters for audit metrics and actual document history.



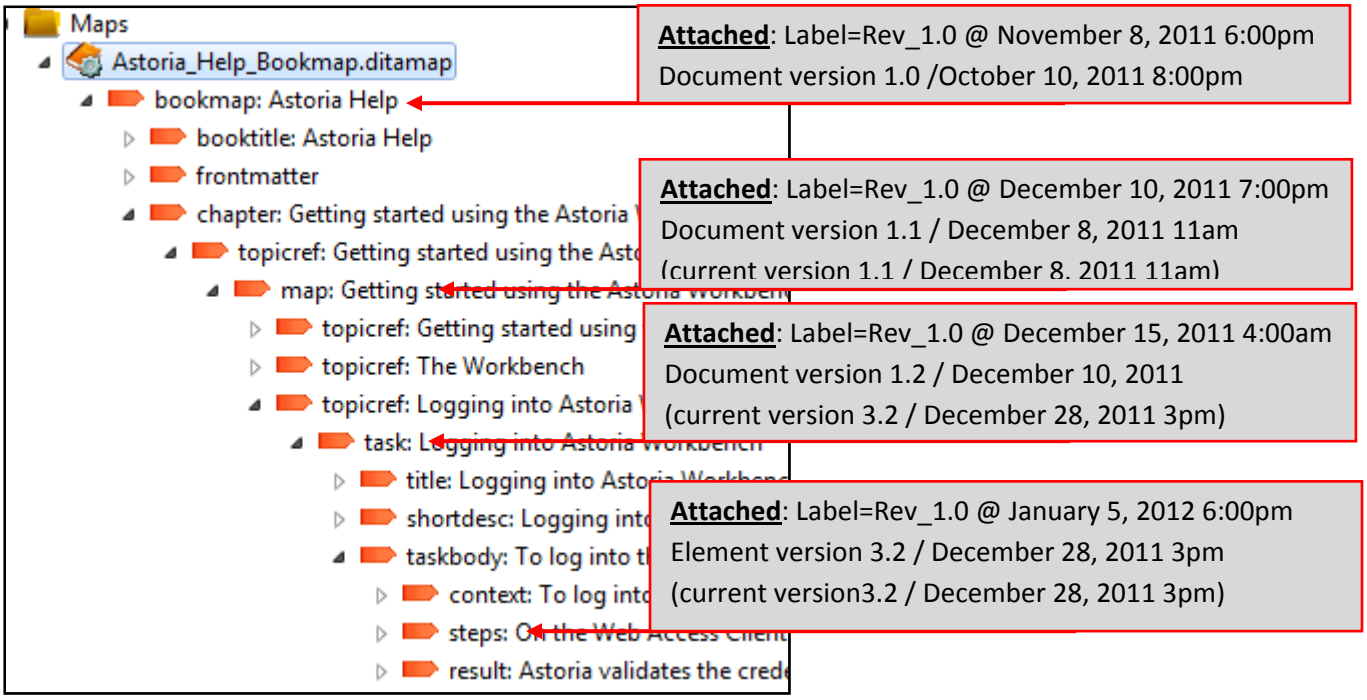
If the changes made for **Rev_1.0** are also needed in the on-going work for the next release, they can be made now on the current version.

Variations for merging changes from different releases

As documents are checked out and changed for specific releases, there are times when a certain set of changes on the current document needs to be merged into one or more of the previous releases.

Accomplishing these merges can take many forms. For instance, if *all* of the changes for a specific release need to be included in the previous release, those changed documents need only to have the same release label used for those previous releases with the addition of the same label with an updated date on those documents.

If a subset of changes needs merging with previous release, the same process is used, only now the label would be on those specific elements where the merged changes are needed. In the following illustration, the <steps> tag in the <task> with the <title>Logging into Astoria Workbench</title> has been changed for the current work on release 3.0. The change to the <steps> are also merged into **Rev_1.0** by adding that label to the <steps> tag, with the result that these changes now apply to both **Rev_1.0** and the future release of **Rev_3.0** – a merging of the two releases with the simple application of a Release Label.



Release Labels and conditional processing

Combining the use of Release Labels with conditional processing (filtering) provides a deeper level of granularity of release management. Each release can include a targeted set of conditions. For example, a bookmap that compiles a dataset for multiple models of a product can be managed on a model by model release schedule.

Release Labels combined with conditional criteria could be used to manage each release for each model separately. Instead of managing the document set with labels like **Rev_1.0**, **Rev_2.0**, or **Rev_2.1**, release management can include **Rev_1.1_models_10, 11, 15, 22**, or **Rev_1.2_brands_GlobalLink, Astoria**.

The addition of conditional processing criteria greatly enhances the focus of release management.

Processing with Release Labels

Release Labels are applied during processing of the dataset for any number of distinct processes. Composition and output, translation packaging, exporting files, or reviewing the files all benefit from this method of release management.

While working on release 3.1, user can still publish or translate approved content contained in release 1.0, 1.1, or 2.0. Updates are made easier for output and translation packaging because only the approved data for each release will be used rather than the current work in progress.

Using release labels for packaging files for translation means that current, non-approved content will not be included in files sent for translation, avoiding wasted time, effort and unnecessary translation costs. Release Labels used during translations evaluate what has changed since the last time each *element* was translated into each specific language, providing the basis for minimizing the amount of data sent for translations

Conclusion

Astoria's method of release management provides the ability to manage *any number* of file and element versions without exposing users to the weaknesses of branch/merge. Release Label management avoids the issue of cloning or branching content, with its attendant issues around re-linking content and managing common content. Also, Release Labels make the merging process trivially easy for ***any number of branches***.