

Using Author-it Variants for Multi-release Software Documentation

Hamish Blunck 2010-03-10

Tested against Author-it v5.3.316.6868

www.HamishBlunck.com



If you use this solution, please register for updates at:

http://www.hamishblunck.com/authorit/aitconfig/index.php/mrd_registration

This document outlines the best practice for using Author-it Variants (introduced in v5.1) for maintaining documentation for multiple releases of software.

The term 'release' is used in this document rather than 'version' to avoid confusion with Author-it's version functionality (which is not used in this solution).

Contents

Prerequisite knowledge	2
Problem.....	2
Solution overview	2
Example.....	3
Solution details	4
Implementing the solution	6
Removing old releases.....	9
FAQ.....	11

Prerequisite knowledge

The document assumes a working knowledge of Author-it Variants. Refer to:

- <http://www.author-it.com/kc/33352.htm> for information about variants.
- <http://www.author-it.com/kc/34200.htm> for an explanation of terminology used.
- <http://www.author-it.com/kc/35144.htm> for an explanation of how Author-it selects a variant when publishing a book if more than one variant matches. This concept is central to the solution.

Problem

Many software products have multiple supported releases deployed on customer sites. Documentation teams are often required to maintain documentation for all supported releases. In addition, the development of content for 'in-development' releases must also be made.

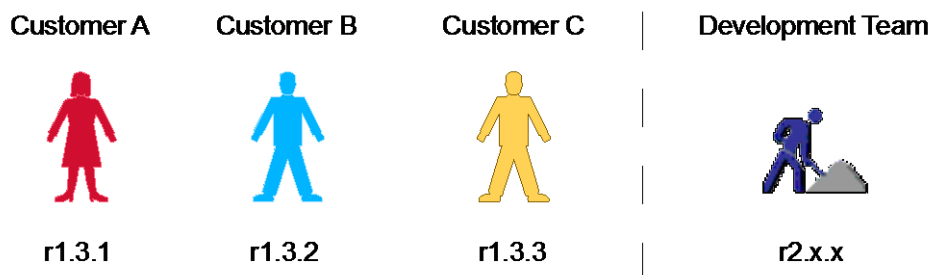


Figure 1 Documenting for multiple releases

Typically, the updates to released software documentation are less intensive than in-development software. For example, updates to released documentation are for bug fixes, minor enhancements, and maintenance.

A methodology for maintaining multiple releases of software documentation is required that has the minimum overhead and complexity for authors.

Solution overview

Author-it's Variants can be used to maintain multiple releases of software documentation allowing documentation to be re-released for earlier releases, while at the same time allowing content development for new releases.

The basic principle behind this solution is to use standard and primary objects for the latest release of the documentation and create variants for any previous releases of documentation. When authoring, if you are updating an object that has changed for a particular release, before updating that object, you create a variant of that object and tag it with the release up until which the content is current. Then you go back to the primary object and update it for the changes for the new release.

From the Author-it Knowledge Centre:

- A **standard** object: Any object without variants. When a book is filtered, it displays the standard objects as usual, along with any variants.
Note: When a variant is created, the standard object becomes a primary object.
 - A **primary** object: Any object that has one or more variants. The primary object maintains the relationship between a book and variant object. When a book is filtered, the primary object is replaced with the variant matching the selected criteria. If the primary object does not have a variant that matches the selected criteria, the primary object is displayed.
 - A **variant** object: A variation of a primary.
-

The solution is a bit counter-intuitive – tagging variant objects with the release up until which they are current, rather than tagging the object variant with the release for which the changes were made. But, if you did the latter, you would need to create a variant for every object for every release – a massive overhead.

Example

As an example, my book has two topics. For the first release, it is easy, the topics are created and no variants are used. When published for the first release, the standard objects are used, as shown in the following diagram.

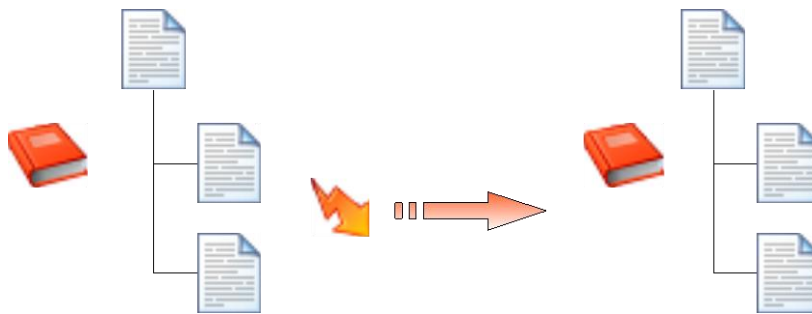


Figure 2 Publishing for the first release

When updating the documentation for the next release, a variant is created and the variant is used to capture the content for the previous releases. The standard and primary objects are updated for the latest release. If an object doesn't change for a release, you do nothing with that object.

When publishing for the latest release, the standard objects are used, as shown in the following diagram.

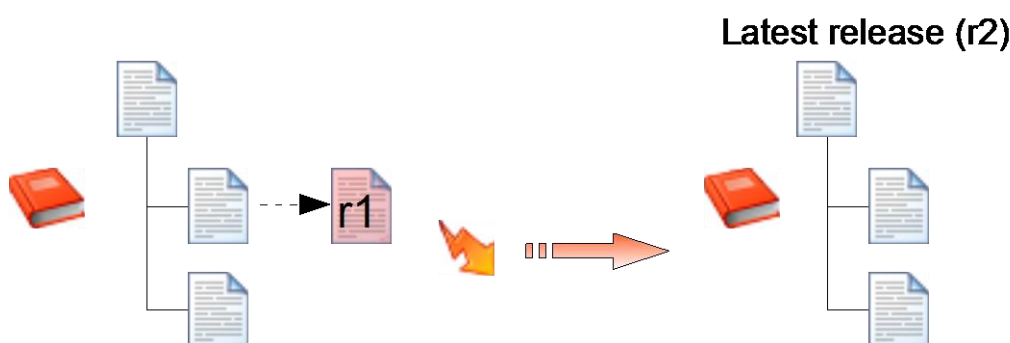


Figure 3 Publishing for the latest release

When publishing for an earlier release, the variant objects are used, as shown in the following diagram.

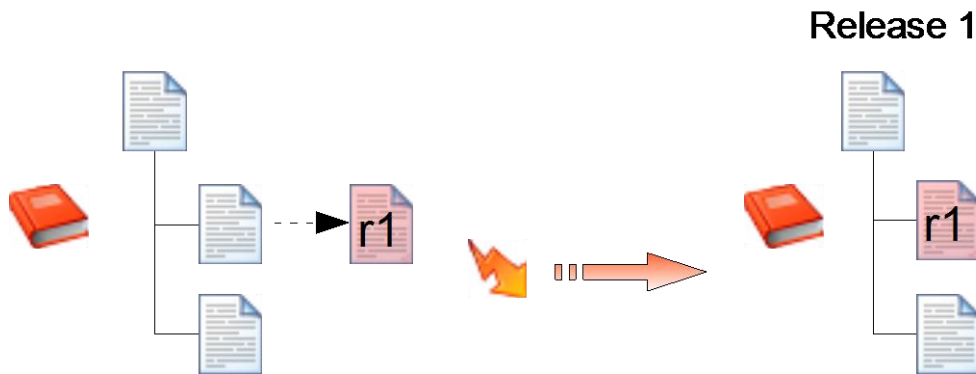


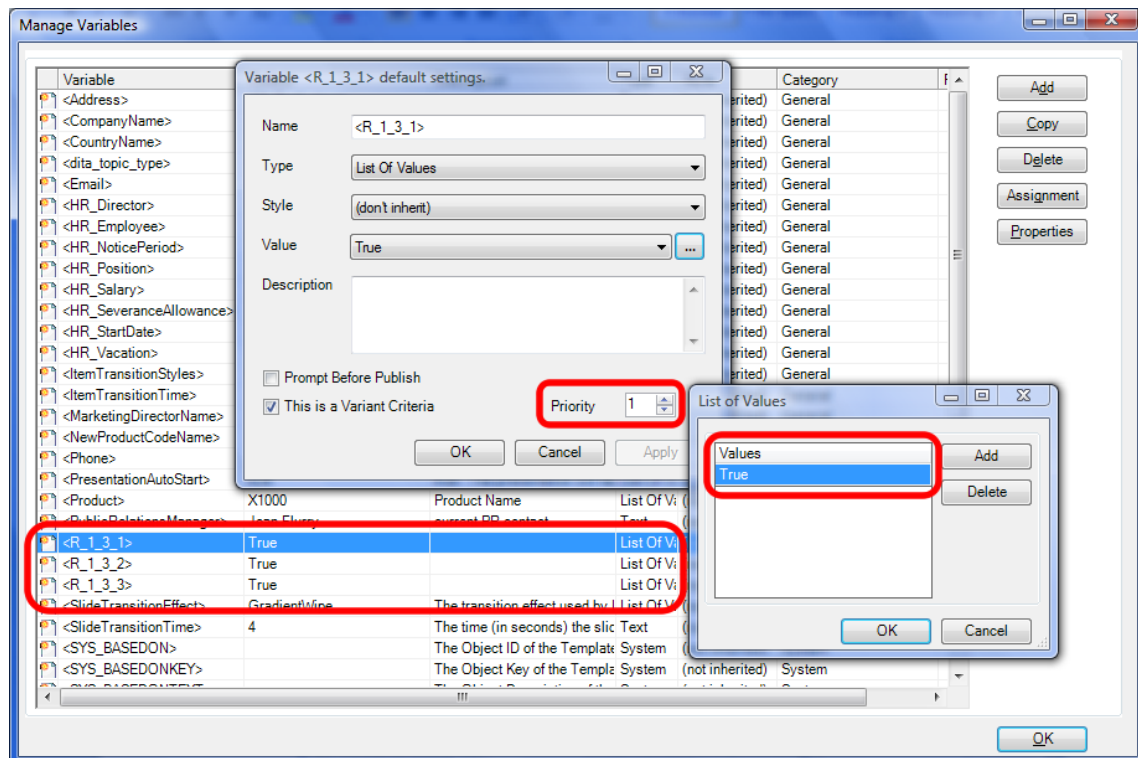
Figure 4 Publishing for a previous release

Solution details

The variant variables

To maintain multiple releases of documentation, a separate variant variable is used for each release of the software; that is, a new variable is created for each release (*not* a single 'Release' variable that has each release in its list of values). Each variable has only a single item in the List of Values: 'True'. When a variant is created, the variant variable for the release up to which the content is current is applied to the object and set to 'True'. For example, in example shown in Figure 4, a topic variant is created and the 'r1' variant variable is associated with the topic and set to 'True'.

The following diagram shows how the variant variables are set up using example releases of r1.3.1, r1.3.2, and r1.3.3.

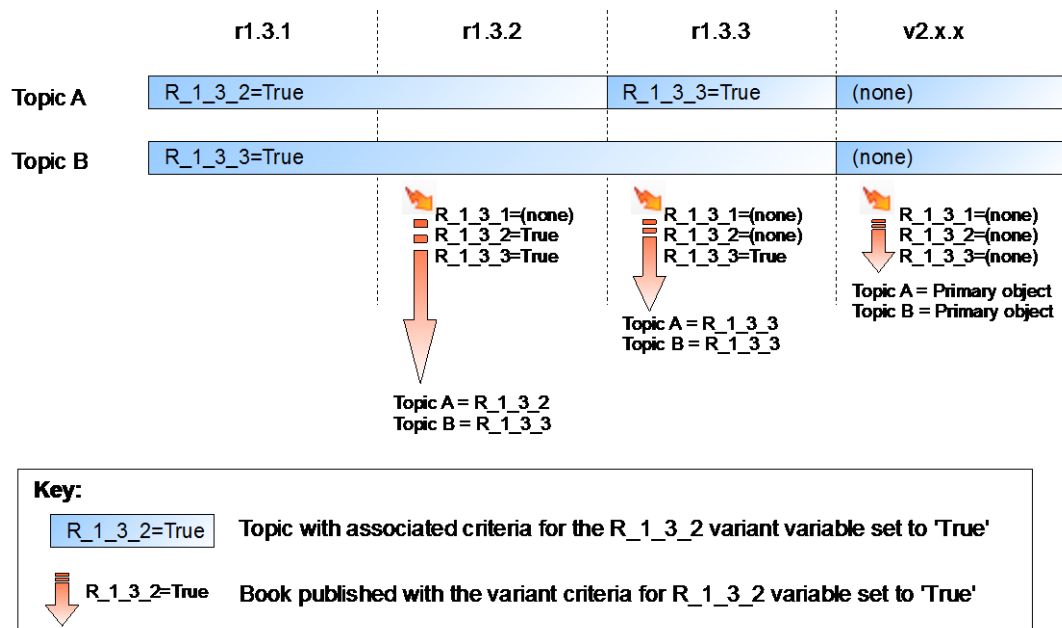


The priority for the earliest release is set to '1', and the priority is incremented for each subsequent release.

Using the variant variables

Standard objects are used until a change is required for a new release. When a change is required, a variant is created and the appropriate variant variable is applied to the object and set to 'True'. The variant variable applied is the variable for the release the object is current up until. For example, if a topic is current up until r1.3.2, the variant variable R_1_3_2=True is used. The primary object is used for the latest release.

As an example, the following diagram shows two topics, Topic A and Topic B, and their variants for each release of the software.



The arrows depict the book being published with different variant criteria selected, and the resultant topics that are published. When a book is published, if you want to publish:

- The latest release of the book, all variant criteria values are set to '(none)'. This means all primary objects will be published – the latest topic variants.
- An earlier release of the book, the variant criteria values for the release you are publishing AND all later releases are set to 'True'. For each object, Author-it will look at all variant objects and select the object with the highest priority (the earliest release of those which is set to 'True'). If no variants exist, the primary object is used.

The same principle applies to editing a book. If you want to edit an earlier release of a book you select all variant criteria for the release you are editing AND all higher releases and set to 'True'.

Implementing the solution

Setting up the variant variables

In the Author-it Administrator, set up a variant variable for each release of the software. Do not set up a variable for the current in-development release.

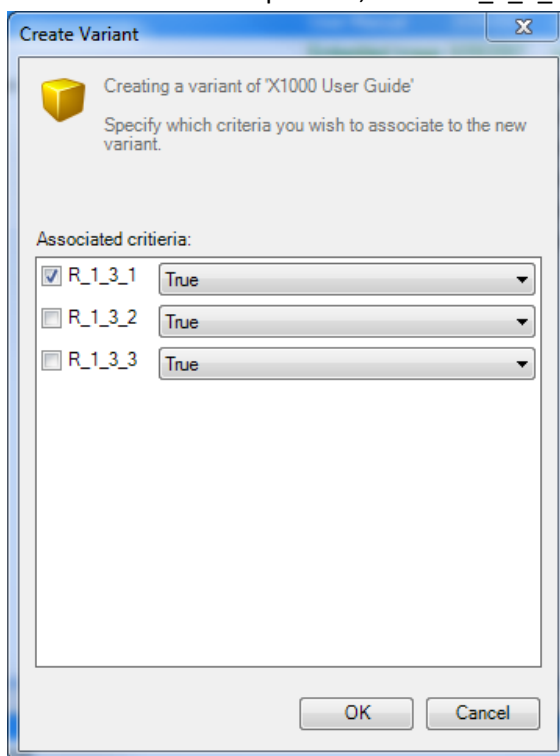
Note: Priority ratings range from 1 (the highest priority) to 99 (the lowest priority).

Use the following properties when creating the variables:

- Type = List of Values
- Value = One item of 'True'.
- Priority = Set to '1' for the earliest release and increment the number for each newer release.
- This is a Variant Criteria = select this checkbox.

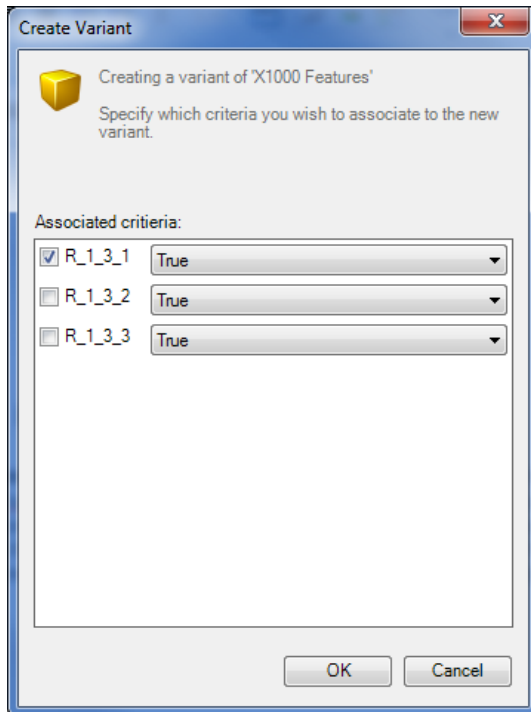
Create a book variant for a new release

1. Create a variant of the book.
2. Set the Variant Variable of the previous release to 'True'. For example, if you have just released r1.3.1 of the product, set the R_1_3_1 criteria to 'True'.



Update topics for the in-development release

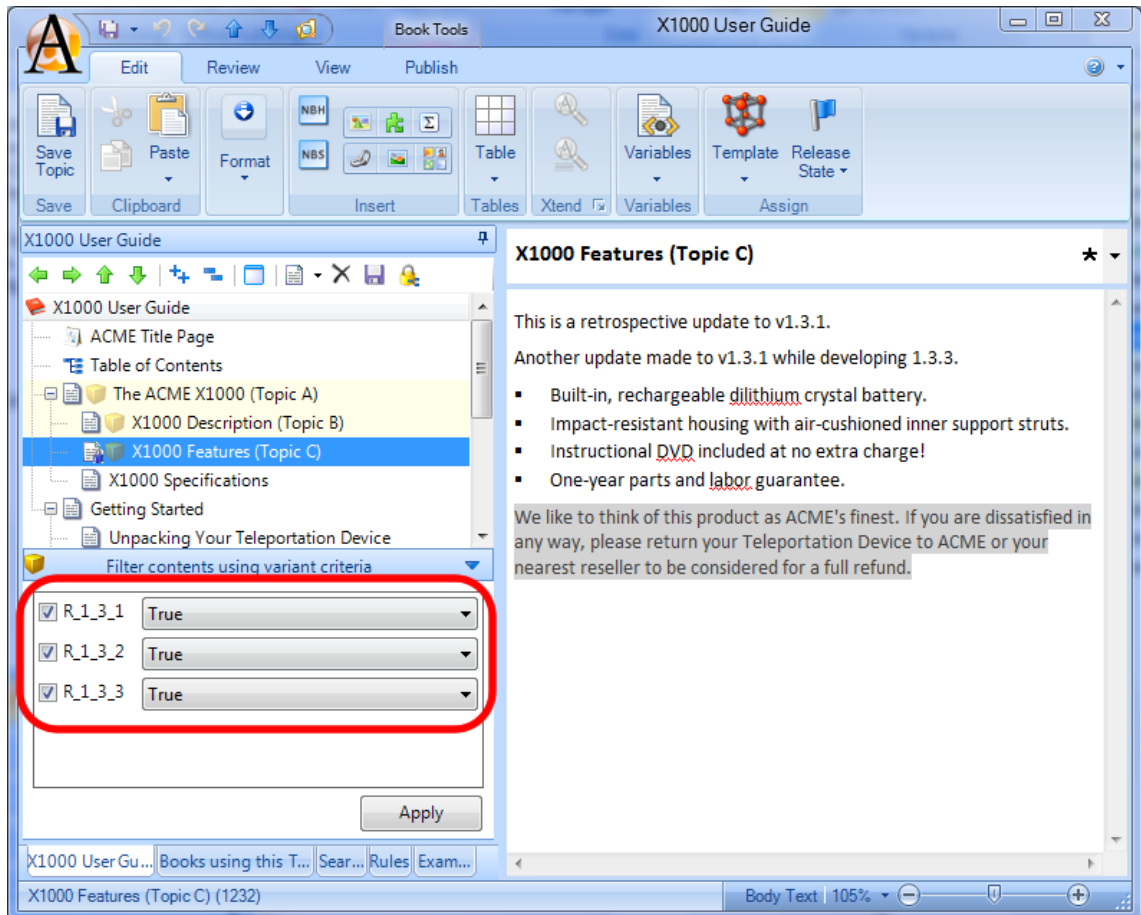
1. Open the primary book object.
2. Create a variant of the topic that has changed for the new release and set the previous release's criteria to 'True'. For example, if you have just released r1.3.1 of the product, set the R_1_3_1 criteria to 'True'.



3. Update the primary topic object with the content for the new release.

Update topics for an earlier release

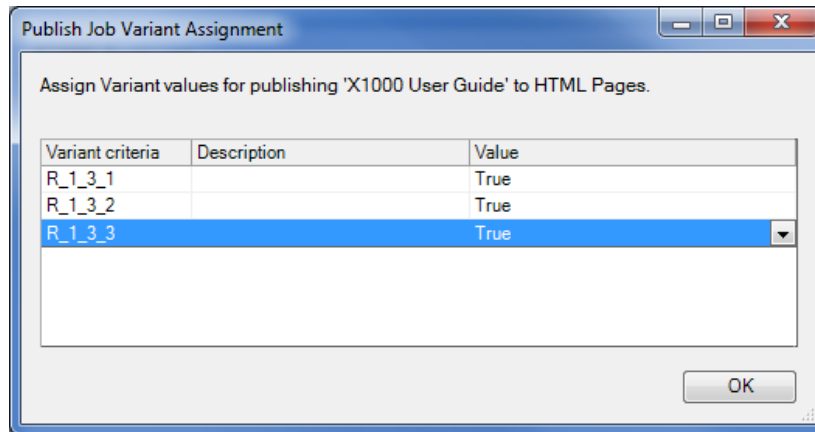
1. Open the book variant for the release you want to update.
2. Filter the contents using variant criteria by selecting all releases from the latest release down to the release you want to update. For example, if updating r1.3.1, select, R_1_3_3, R_1_3_2, and R_1_3_1.



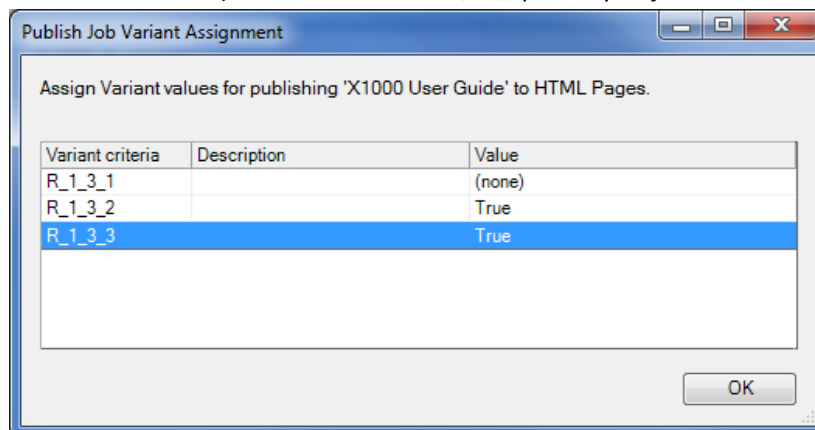
3. If required, create a topic variant (if one doesn't already exist for that release).
4. Make the changes to the topic variant.
If the change applies to other releases (variants) of the topic, you must manually make these changes in the other topic variants. For example, if a new feature was added in an earlier release and has been included in all later releases, you need to manually update the topic variants with this change.

Publishing the book

1. Open the book variant for the release you want to publish the book for (or the primary object, if for the current release).
2. On the Publish Job Variant Assignment window, to publish:
 - The latest release of the book, set all variant criteria values to '(none)'. This means all primary objects will be published – the latest release of the book.



- An earlier release of the book, set all variant criteria values for the release you are publishing AND all later releases to 'True'. For each object, Author-it will look at all variant objects and select the object with the highest priority (the earliest release of those set to 'True'). If no variants exist, the primary object is used.



Removing old releases

You can remove old releases from Author-it if they are no longer required; that is, you no longer need to maintain or publish the old releases.

Tips:

- Backup the library before you complete this procedure so you can recover a lost version if necessary.
- Using the Author-it Administrator re-index the library using the **Maintenance > Re-index** Search menu option. This makes sure the search will find all the required variants.

To delete a release:

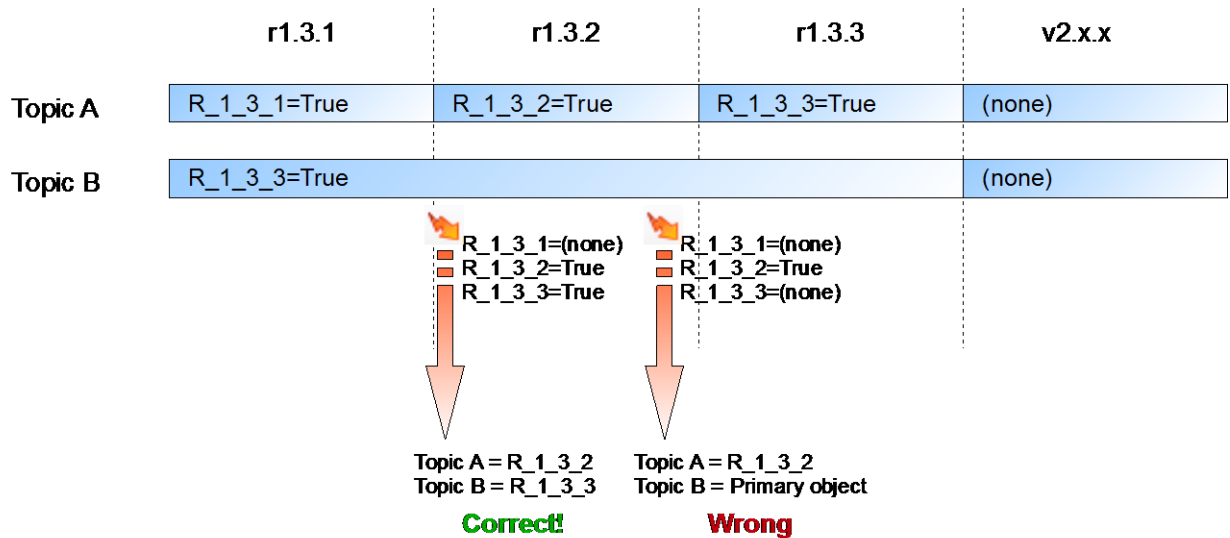
1. Create a release state that will be used to tag the variants to be removed; for example 'To Remove'.
2. Always start with the earliest release. It is not recommended to delete a release which is not the earliest. Search for all variant objects for the release. For example, search for '<r_1_3_1>'.
3. Sort the search results by the **Variation** column so you are sure that you are only picking up variant objects rather than an object that has the variant criteria you searched for.
4. Select the objects and change their release state to the release state created in step 1. This is a safeguard in case you clear the search after you have removed the variant criteria – so you can locate all of the variant objects.
5. Save the objects to XML. This will allow you to re-import the objects at a later time, if required.
6. Select all objects and click the **Remove** button in the **Manage > Variants** ribbon group.
7. Delete all of the objects.
8. In the Author-it Administrator:
 - a. Delete the release variable.
 - b. Reassign the priorities for the other release variant variables so the earliest starts at '1'.
9. Close and re-start Author-it.

FAQ

When updating an older release, why do I need to set the variant criteria for all releases from the latest release down to the release I want to update? Why can't I just select the variant criteria for the specific release I want to update?

You cannot just select the release you want to update because you have not necessarily created a variant for every object for every release. Remember, that with this solution, you only create a variant if the object has changed for a particular release.

For example, take the following scenario.



Topic A has a variant for each release. Topic B only has a variant for r1.3.3.

If I wanted to update the documentation for r1.3.2, and you only set the R_1_3_2 variant criteria to 'True', for Topic A, Author-it would display the correct topic to update. However, Author-it would display (and you would edit) the primary object for Topic B. This is because, there is no R_1_3_2=True criteria for Topic B – If Author-it can't find an exact match, it will choose the primary object.

If you selected both R_1_3_3=True and R_1_3_2=True, then Author-it would select the r1.3.3 object for Topic B to edit, which would be correct. (Unless of course, Topic B needs to be different for r1.3.2 and r1.3.3, in which case you would need to create a new variant.)

The same principle applies to publishing too. You need to set the variant criteria for all releases from the latest release down to the release I want to publish.