

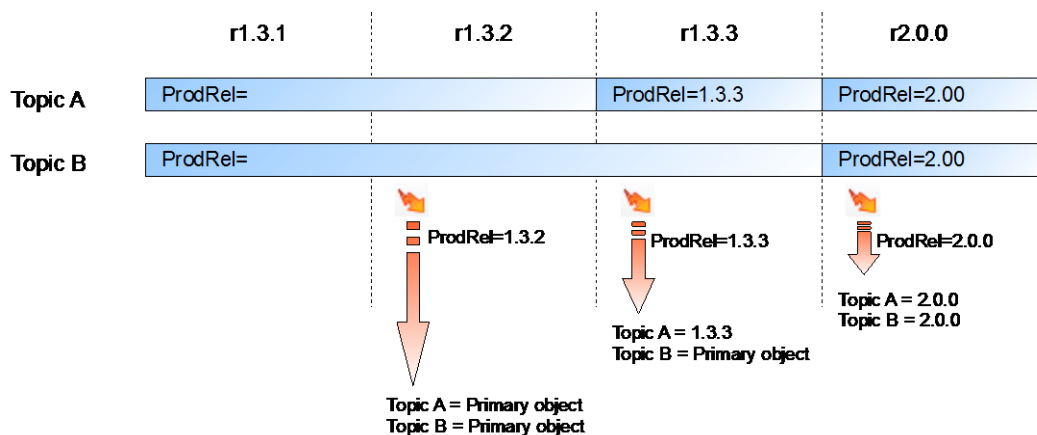
## Alternative solutions

The following sections outline two alternative solutions that were considered, but dismissed.

### Alternative solution 1

- One variant variable is used (for example, ProdRel), with an item in the List of Values for each release, for example, r1.3.1, r1.3.2, r1.3.3, r2.0.0.
- Standard objects are used (objects without variants) until a change is required for a new release. When a change is required, a variant is created and the variant criterion is applied to the new variant (the latest release). For example, ProdRel=r1.3.2.
- When the book is published, the appropriate release is selected as the variant criteria value and the book is published for that release.

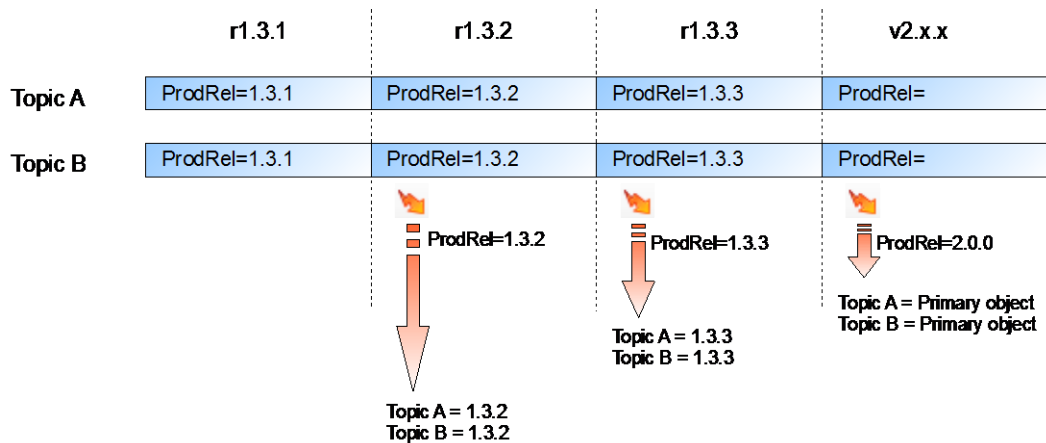
The following diagram shows two topics, Topic A and B, and their variants for each release of the software. The arrows depict the book being published with different variant criteria selected, and the resultant topics that are published.



The main disadvantage of this solution is that a variant must be created for *each object*, for *each release* after an initial change has been made. Otherwise, the incorrect variant may be published. This is a huge overhead for authors and is prone to mistakes. For example, the content of Topic A may not have changed between r1.3.3 and 2.0.0, but a new variant must be created with ProdRel=2.0.0, otherwise the primary topic object would be published when publishing the r2.0.0 book (without the changes made in r1.3.3).

### Alternative solution 2

The second solution is very similar to Solution 1, except, variants are used for the old releases, and the standard object is used for the change for the latest release. The variant criterion applied to the object is the release number that the object applies to.



This solution has even more overhead than Solution 1 because a new variant must be created for each object for each release, irrespective of whether the topic has changed from one release to the next.

### Discussion

The main drawback for how Author-it currently implements variants is that when publishing you can only specify an *exact* match with variant criteria. This means for Solutions 1 and 2, you need to have a variant for each release for each object (so that Author-it will always select the correct variant). This is a huge overhead for authors.

If you were able to specify 'the highest variant value up to an including' a variant value, Solution 2 could be used without having to create a variant for each release for each object.

Based on this limitation, the main solution discussed in this document is the best practice. This solution has the advantage over the other solutions in that variants are only created when an object changes for a release.

### Wish List

- Publish Job Variant Assignment window: ability to specify the default value for each variant (maybe taking the default value from the Manage Variables window in the Administrator).