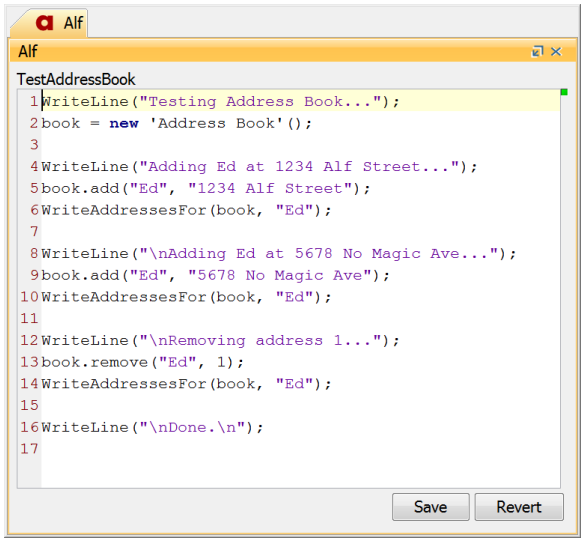
The Alf editor

The Alf editor window

The Alf editor window provides a general way to edit the Alf code of any element with an Alf body. If your project was created using the Alf project template, then the Alf editor window will already be open and docked in the lower left window pane. But, if it is not open, you can open it by selecting **Windows > Alf** (the window will initially appear at the bottom of the main MagicDraw window, but it may be moved and docked like other MagicDraw windows). The Alf editor window remains open and in the same position for a given project, even when the project is closed and opened again. For opaque behaviors, expression and actions, associated Alf code may also be edited as the body of those elements (for more details on editing the Alf bodies of these kinds of model elements, see the child pages of this page).

To edit Alf code in the Alf editor window, open the window (if it is not already open) and select the element whose Alf body is to be edited. If the element has an Alf body (or if it is possible to add a new Alf body to it), then the Alf code appears in the window. As shown in the sample image below, the code is displayed with keywords and syntactic elements highlighted in different colors.



The Alf editor window

The buttons at the bottom right of the window have the following functions.

Button name	Description
Save	Save changes that have been made to the Alf body being edited. If, after making changes to the Alf code, you select a different element, then your changes will be automatically saved, even if you have not pressed the **Save** button.
Revert	Revert the contents of the window to the last saved version of the Alf body. An changes made since the last save will be lost.

Alf compilation errors

When you save Alf code from the editor, if the code has no errors, then it is automatically compiled so that it becomes executable. If the code has errors, however, then it can still be saved as text, but it cannot be compiled.

> ⚠ An Alf body that is saved with compilation errors may have been previously compiled successfully. In this case, the executable behavior of the element with which it is associated will still reflect that generated from the last successful compilation.

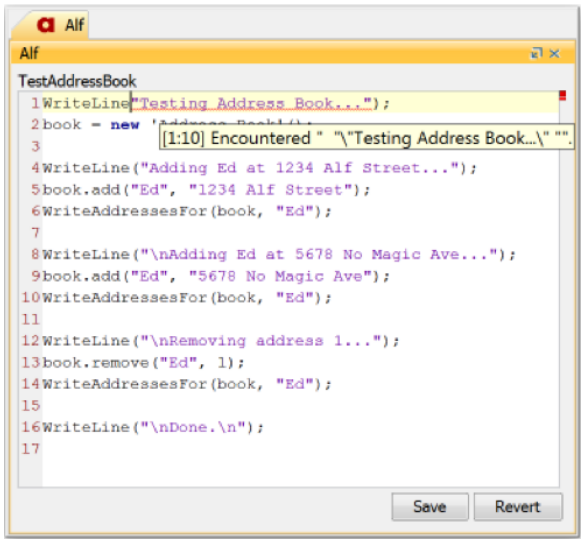
The Alf editor will detect errors in Alf code as it is being edited. Detected errors are identified by underlining the relevant text in red and showing a red marker to the left of any line containing an identified error. Alf code may have two kinds of errors: *syntax* errors, which result from a failure to parse the text being edited as Alf code, and *constraint violations,* which are violations of the semantic checks made on Alf code that has been parsed successfully.

Table of Contents

Related pages

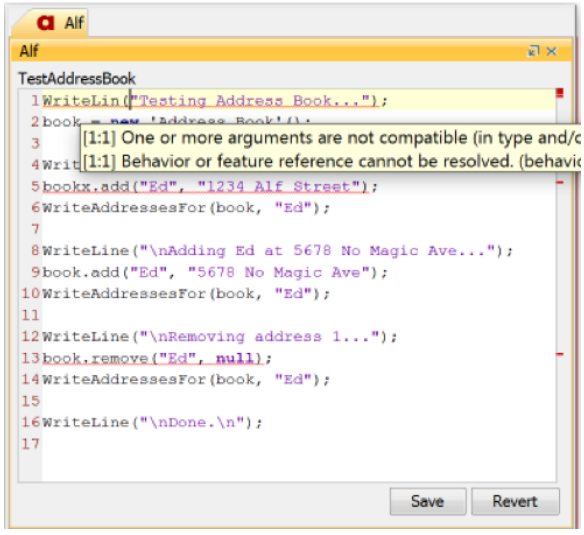
- Working with Alf
  - Using Alf to define Behaviors
  - Using Alf in Class models
  - Using Alf in State Machine models
  - Using Alf in Activity models
- The Alf compiler

The Alf editor abandons parsing the text on the first syntax error. Therefore, text with a syntax error will have only one marker, at the point at which the parse failed. The image below shows an example of text with a syntax error. Hovering the cursor over the marked text (or the marker to the left of the line) shows the cause of the error.



A syntax error

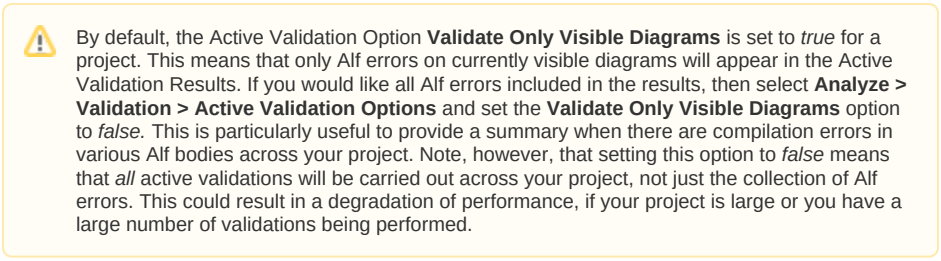
If the text being edited parses successfully, then the Alf editor runs a series of *constraint checks* (so called because these checks are formally specified in the Alf standard as constraints on the abstract syntax tree of parsed Alf code). As shown in the example below, there may be multiple constraint violations, with the relevant text for each of them being identified. Hovering the cursor over the marked text (or the marker to the left of the line) shows what constraints have been violated.



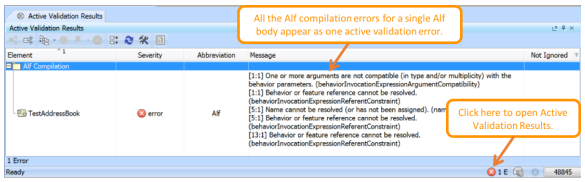
Constraint violations

If you save Alf code with errors, then these errors will also be recorded in the Active Validation Results window (as shown below).

> Depending on your Active Validation environment options, it may take a couple of seconds after you save your Alf code before the error notifications appear in the Active Validation Results (or before they disappear, once they are corrected)



By default, the Active Validation Option **Validate Only Visible Diagrams** is set to *true* for a project. This means that only Alf errors on currently visible diagrams will appear in the Active Validation Results. If you would like all Alf errors included in the results, then select **Analyze > Validation > Active Validation Options** and set the **Validate Only Visible Diagrams** option to *false*. This is particularly useful to provide a summary when there are compilation errors in various Alf bodies across your project. Note, however, that setting this option to *false* means that *all* active validations will be carried out across your project, not just the collection of Alf errors. This could result in a degradation of performance, if your project is large or you have a large number of validations being performed.

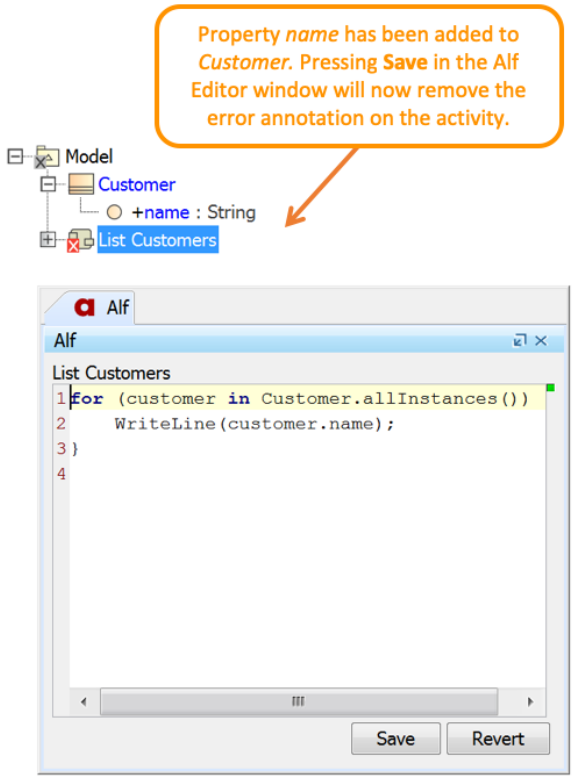


Alf compilation errors in Active Validation Results

Removing error annotations

Sometimes, an element will remain marked with an Alf compilation error annotation, even though, when you look at the Alf code associated with it in the Alf Editor, there are no errors. For example, if you enter Alf code that references a property that does not exist, this will be marked as an error. If you later create a property consistent with the reference in the Alf code, the original error annotation remains, because no dependency of the Alf code on the property could be recorded before the property existed. However, if you then look at the Alf code in the Alf Editor, it will not have any errors.

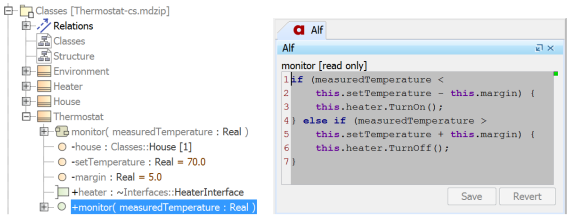
Previously, it was generally necessary to do a project build to remove such error annotation. Simply trying to save the (now correct) Alf code from the editor had no effect, unless you made some change to the text. In the Alf Plugin 19.0 SP2, however, saving valid Alf code from the editor will now always remove any error annotation on the corresponding model element, even if you have made no changes to the text.



Saving from the Alf Editor to remove an error annotation

Read-only Alf bodies

Sometimes an Alf body will be attached to an element that is editable, such as when it is in a read-only used project or a part of a Teamwork Cloud project that is not locked for edit. In this case, the Alf Editor will still show the Alf code for the element, but on a gray background to indicate that the code is not editable. If the element becomes editable (for instance, by locking it for edit in Teamwork Cloud), then the Alf Editor will allow the code to be edited.



Viewing the Alf body of a read-only element