Using Alf for State Behaviors

A State in a State Machine can have an entry Behavior (which is executed when the State is entered), an exit Behavior (which is executed when the State is exited) and/or a do-activity Behavior (which is executed asynchronously while the State is active). You can us the Alf editor to create an Alf body for, or edit the existing Alf body of, any one of these kinds of Behaviors.

Related pages

- The Alf editor
- The Alf compiler

To create and Alf body for a State behavior

1. Open the Specification window for the State (from the Model Browser or from a State Machine diagram with the State on it).
2. In the **Entry**, **Do Activity** or **Exit** section, as appropriate, click on the **Behavior Type** property and select **Activity** or **Opaque Behavior**. Enter a name for the new Behavior into the corresponding **Name** field.

   ⚠ It is actually optional to enter a name for the new Behavior. If the Behavior is an Opaque Behavior, and the **Opaque Behavior Display Mode** symbol property of the State is set to Body, then the Alf code will be displayed for the Behavior within the owning State on a State Machine diagram. However, if the Behavior is an Activity, this symbol property does not apply, and it is always the Activity name that is displayed. Therefore, if you use an Activity for a State Behavior, you should always give it a name, so that this can be shown in its representation on a State Machine diagram.

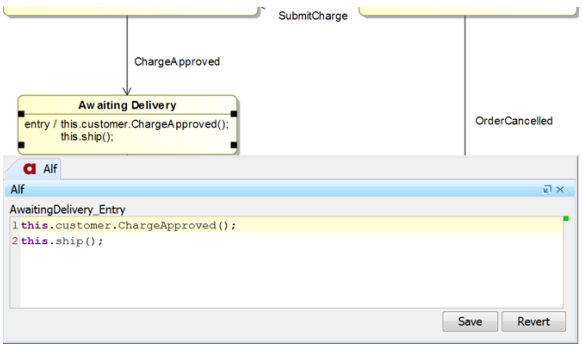
3. Close the Specification window.
4. Select the Behavior newly created under the State (either in the Model Browser or as it appears on a State Machine diagram) and open the Alf editor window (select **WIndows > Alf**), if it isn't already open.
5. Enter the Alf code for the Behavior and press **Save** to compile and save it.

   ⚠ The Alf code for a State Behavior can access attribute values of the context Classifier of the State Machine containing its State (using Alf *this* expressions). It can also access the data in the Event occurrence that triggered its execution (e.g., the attribute data of a received Signal).

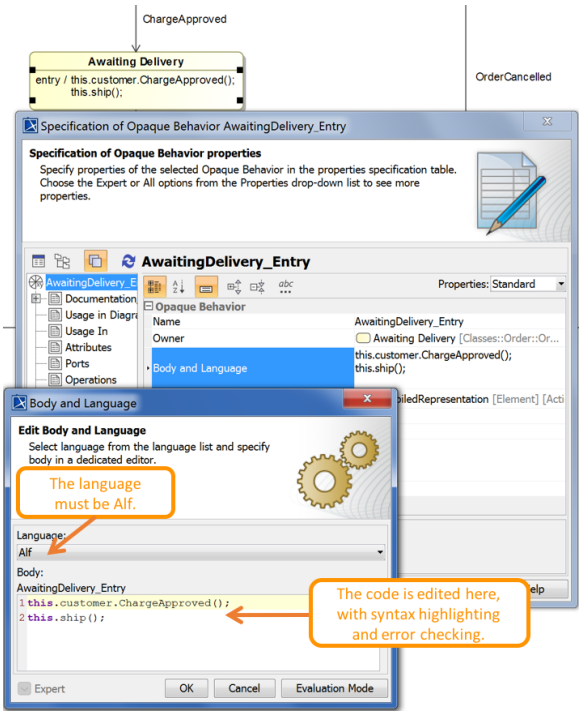
To edit the Alf code for a State Behavior

- Select the Behavior (in the Model Browser or as represented on a State Machine diagram) and open the Alf editor window (select **Windows > Alf**), if it isn't already open. The Alf code will appear in the window.

  ⚠ To edit a State Behavior on a State Machine diagram, be sure to select the specific Behavior that you want to edit (entry, do or exit), and not on the State itself.



If you create a State Behavior as an Opaque Behavior, then Alf code for it can also be entered or edited directly in the **Body and Language** property for it in the Specification window for the State. If you open the Edit Body and Language window, then the Alf code may be edited just as in the Alf editor window. However, instead of **Save** and **Revert** buttons, this window has **OK** and **Cancel** buttons, either of which will close it. If you press **OK**, then the text is saved, and, if it has no errors, compiled. If you press **Cancel**, the Alf text is *not* saved.



Editing Alf code for a State Behavior in an Opaque Behavior body

To be recognized as Alf code, the **Body** of an Opaque Behavior must have the **Language** Alf. If your project was created using the Alf project template, then Alf will be the default language. Otherwise, select Alf from the **Language** menu (you may have to scroll upwards in the menu to find the selection for *Alf*).

To make Alf the default language for Opaque Behaviors

1. Select **Options > Project**.
2. On the left, under **Default model properties**, select **Opaque Behavior**.
3. Click on the **Language** field and then on the small + button on the right.
4. Enter *Alf* (spelled and capitalized in exactly that way) and click on **OK**.
5. Click on **OK** to close the options window.

If the **Opaque Behavior Display Mode** symbol property of a State is set to Body, then any State Behaviors that are Opaque Behaviors can also be edited directly on a State Machine diagram showing the State and its behaviors. Since such editing does not use the Alf editor, no parsing or constraint checking happens while Alf code is being edited. The code is compiled once you finish editing it, and, if there are compilation errors, they will be recorded in an error annotation on the Opaque Behavior.