# The Alf compiler

The *Alf compiler* is the part of the Alf plugin that carries out the compilation process. The Alf compiler is automatically invoked when you save Alf code into your model (unless you turn off automatic building; see <u>Environment options</u>). The activity model resulting from an Alf compilation is also stored in your model. Exactly where it is stored depends on which kind of element the Alf code is the body.

## Activity compilation

When the Alf compiler processes the Alf body of an Activity, the resulting Activity model is saved as the Activity Nodes and Edges owned within that Activity. The Alf text itself is stored in a Comment element owned by the Activity, which has the *TextualRepresentation* stereotype applied. (See also Using Alf to define Activities and Using Alf for Operation methods.)

Image: Strate	Address Book		
<pre>     Boy/TerAdressBook     Withins CfreeAdressBook*;;     Boy/TerAdressBook     Withins CfreeAdressBook*;     Boy/TerAdressBook*;     Boy/TerAdressBook*;     Boy/TerAdressBook*;     DerssionStament@13755     Boy/Statement@13755     Boy/Statement@145275     BoyresionStatement@151504f     BoyresionStatement@315051     Boy, bok = new '</pre>	E Entry	Alf	a × ا
<pre>2 book = new 'Address Book'(); 3</pre>	E- TestAddressBook	TestAddressBook	
<pre>3     (*********************************</pre>	Body(TestAddressBook)	1WriteLine("Testing Address Book");	
<pre></pre>	Interpretations	2 book = new 'Address Book'();	
<pre>book.add("Edf," 12/3 Alf Street");</pre>	ExpressionStatement@1c371f55		
<pre>Generation of the set of the</pre>	Interpretations		
Image: Solution of the solution	E- Call(WriteLine):WriteLine		
B: O Value(* Done.')       BWriteLing(*\nAdding Ed at 5.678 No Magic Ave');         B: ExpressionStatement(\$15227       BPootv.ad('Ed', '', '', ''S78 No Magic Ave');         B: ExpressionStatement(\$15274'       DWriteAddressesFor (book, "Ed');         B: ExpressionStatement(\$154024'       1         B: ExpressionStatement(\$154024'       1         B: ExpressionStatement(\$154024'       1         B: ExpressionStatement(\$154024'       1         B: ExpressionStatement(\$164024'       1         B: ExpressionStatement(\$164024'       1         B: ExpressionStatement(\$124024'       1	E- 0 Tuple@1d68821d		
Image: Supersoins/Statement@164227f       9book.add("Ed4", "5c7" No Kapic Ave");         Image: Supersoins/Statement@6415774       9book.add("Ed4", "5c7" No Kapic Ave");         Image: Supersoins/Statement@6415744       11         Image: Supersoins/Statement@6415744       11         Image: Supersoins/Statement@930c2511       13book.remove("Ed", 1);         Image: Supersoins/Statement@930c2511       13book.remove("Ed", 1);         Image: Supersoins/Statement@930c2511       13book.remove("Ed", 1);         Image: Supersoins/Statement@930c2511       15book.remove("Ed", 1);         Image: Supersoins/Statement@93055350       15         Image: Supersoins/Statement@93055350       12         Image: Supersoins/Statement@93055350       12         Image: Supersoins/Statement@93055350       12         Image: Supersoins/Statement@93055350       12         Image: Supersoins/Statement@9305350       12         Image: Supersoins/Statement@9305350       12         Image: Supersoins/Statement@9305350       12         Image: Supersoins/Statement@9305350       13         Image: Supersoins/Statement@9305350       14         Image: Supersoins/Statement@9305350       15         Image: Supersoins/Statement@9305350       15         Image: Supersoins/Statement@9305350       15         Im	i → → Value(" Done. ")		
Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Ed");         Image: Construct Statement Strip: 54       10 WriteAddressesFor (book, "Address Strip: 54         Image: Construct Strip: 54       10 WriteAddressesFor (book, Address Strip: 54         Image: Construct Strip: 54       10 WriteAddresses For (book, Address Strip: 54         Image: Construct Strip: 54       54         Image: Construct Strip: 54       54         Image: Construct Strip: 54       54         Image: 54       54         Image:	E ExpressionStatement@1fe3272f		·
ExpressionStatement@312b0dfe     12WriteLine("\nDemoving address 1");     11book.recover(tzim, 1);     11book.recover(tzim,	ExpressionStatement@5f1f7574		
BopressionStatement@392c53ta     13boot.rearvow("Ed", 1);     BopressionStatement@392c54bd3     BopressionStatement@701acc64     BopressionStatement@21297c1     BopressionStatement@375535     BopressionStatement@	E- ExpressionStatement@6a1517e4	11	
ExpressionStatement@34cb304.3     ExpressionStatement@34cb304.3     ExpressionStatement@2401304.3     ExpressionStatement@2401307.4     ExpressionStatement@240137.4     ExpressionStatement@370537     ExpressionStatement@370533     ExpressionStatement@37053     ExpressionStatement@37053     ExpressionStatement@3705     ExpressionStatement@3	ExpressionStatement@13ba0dfe		
Cytessurfactamenteg701acc64     GeressionStatementeg701acc64	ExpressionStatement@39c25d1a		
CopressionStatement@201957     Constantement@201957     Constantement@201957     Constantement@2129747     Constantement@201957     Constante			
CopressionStatement(#01307)     CopressionStatement(#01307)     CopressionStatement(#030875330     Cocate Save Revert     Cocate Save Revert     Save Revert     Cocate Save Revert     Save Save Revert     Save Save Revert	ExpressionStatement@701acc64		
OperesionStatement@2129747     OperesionStatement@375530			
Concerning and a standard a standard a standard and a standard a standar		£.7	Ψ.
WriteLine("Testing Address Book"); book = new ' «TextualRepresentation»  WriteAddresser		Create Save	Revert
E: WitteAddressefer( book : Address Book, name : String )			
The code text itself is	E − B WriteAddresses pr( book : Address Book, na	ame : String )	
The code text itself is			
The code text itself is	<u> </u>		
	The code text itself is		

#### **Table of Contents**

- Activity compilation
- Opaque Behavior and
- Opaque Action compilationOpaque Expression
- compilationThe \$\$Template Bindings
- package

#### Activity compilation

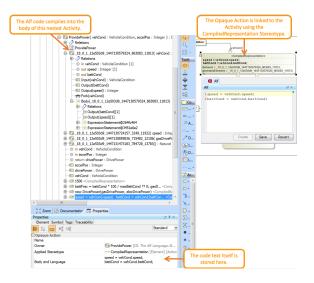
The compilation of an activity may also result in certain auxiliary elements, such as Instance Specifications, that cannot be not stored within the Activity. These are generally inserted into the nearest Package containing the Activity (which may be the top-level Model). Also, in certain cases, template instantiations are saved in a special package called *\$Template Bindings* (described below).

### **Opaque Behavior and Opaque Action compilation**

When the Alf compiler processes the Alf body of an Opaque Behavior or Opaque Action, it creates a corresponding Activity in which to store the result of the compilation. For an Opaque Behavior, the generated Activity is saved as a child of the Opaque Behavior. For an Opaque Action, the generated Activity is saved as a child of the Activity containing the Opaque Action. In either case, the Activity is also linked to the original Opaque Behavior or Action by applying the *CompiledRepresentation* stereotype. When the Opaque Behavior or Action is later executed using Magic Model Analyst, it is the linked Activity that actually provides the executable behavior.

The Alf code con body of this nes		The Opaque Behavic Activity usi CompiledRepresenta	ing the
WriteAddress of fir (book : A     WriteAddresseFor(boo     WriteAddresseFor(boo	k : Address Book [1], r pok [1] ss Book		ਹ ×
		WriteAddressesFor 1WriteLine("Addresses for " 2for (address in book.lookup 3 WriteLine(address); 4) 5	(name)) {
Call(WriteLing) Call(W	0421	Crea	te Save Revert
Com Documentation	Properties	17. # X	
Element Tags Traceability		C * ^	
		Standard 💌	
Opaque Behavior			
Name	WriteAddres	sesFor	
Owner	Model		
Body and Language	for (address	ddresses for " + name + ":"); in book.lookup(name)) { (address);	The code text itself stored here.

**Opaque Behavior compilation** 

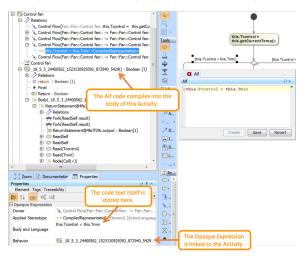


**Opaque Action compilation** 

# **Opaque Expression compilation**

When the Alf compiler processes the Alf body of an Opaque Expression, it creates a corresponding Activity in which to store the results of the compilation. This Activity is generally saved in the nearest Classifier containing the Opaque Expression. If the Opaque Expression specifies the guard on a Transition, then the generated Activity will be saved as a child of the State Machine containing the Transition. And, if the Opaque Expression specifies the guard on an Activity will be saved as a child of the Activity Edge, then the generated Activity will be saved as a child of the Activity Edge.

In all cases, the Activity is referenced in the **Behavior** property of the original Opaque Expression. When the Opaque Expression is later evaluated using Magic Model Analyst, it is the **Behavior** Activity that is actually executed in order to provide the value of the Expression.





## The \$\$Template Bindings package

The \$\$Template Bindings package is a special package used by the Alf compiler to save instantiations of template Classifiers that result of explicit template bindings in Alf code (e.g., Set<Integer>). These instantiations are stored in a common place, so that similar bindings in Alf code across your model can be compiled to references to the same instantiation. For example, a binding of a standard Alf collection class, such as Set<Integer>, results in the generation of an instantiated class in the \$\$Template Bindings package with a long encoded name similar to \$\$Model\$\$Alf\$\$Library\$\$CollectionClasses\$\$Set\_\$\$Mod el\$\$'UML Standard Profile'\$\$'UML2 Metamodel'PrimitiveTypes\$\$Integer\_. If you use Set<Integer> in more than once in Alf code in your model, it will still result in only the one instantiated class, and the Activity models compiled from your Alf code will all refer to this class.

The \$\$Template Bindings package should only appear if you use template bindings in your Alf code. It is automatically created the first time such a binding is compiled in your model.

You should never manually alter any of the contents of the \$\$Template Bindings package. (1)

When you delete the code for a template binding with an Alf body, the instantiated class generated from that Alf code is *not* automatically deleted from the *\$\$Template Bindings* ⚠ package, since that class might also be referenced elsewhere in your model. Thus, if you use templates in your Alf code, it is possible, over time, for the \$\$*Template Bindings* package to accumulate contents that are no longer useful. If necessary, this can be cleaned up by doing a *clean build* of your project.

E- A Model
🖨 🛅 \$\$TemplateBindings
IE−
\$\$Model\$Alf\$Library\$CollectionClasses\$Collection_Model\$'UML Standard Profile'\$'UML2 Meta
\$\$Model\$Alf\$Library\$CollectionClasses\$Impl\$Set_Model\$'UML Standard Profile'\$'UML2 Meta
and detailed the second of the second s

amodel'\$PrimitiveTypes\$Intege model'\$PrimitiveTypes\$Intege

\$\$Model\$Alf\$Library
Collection<Integer>
Set<Integer>
Set<Integer>
Set<Integer>

The \$\$Template Bindings package, showing the instantiation of Set<Integer>