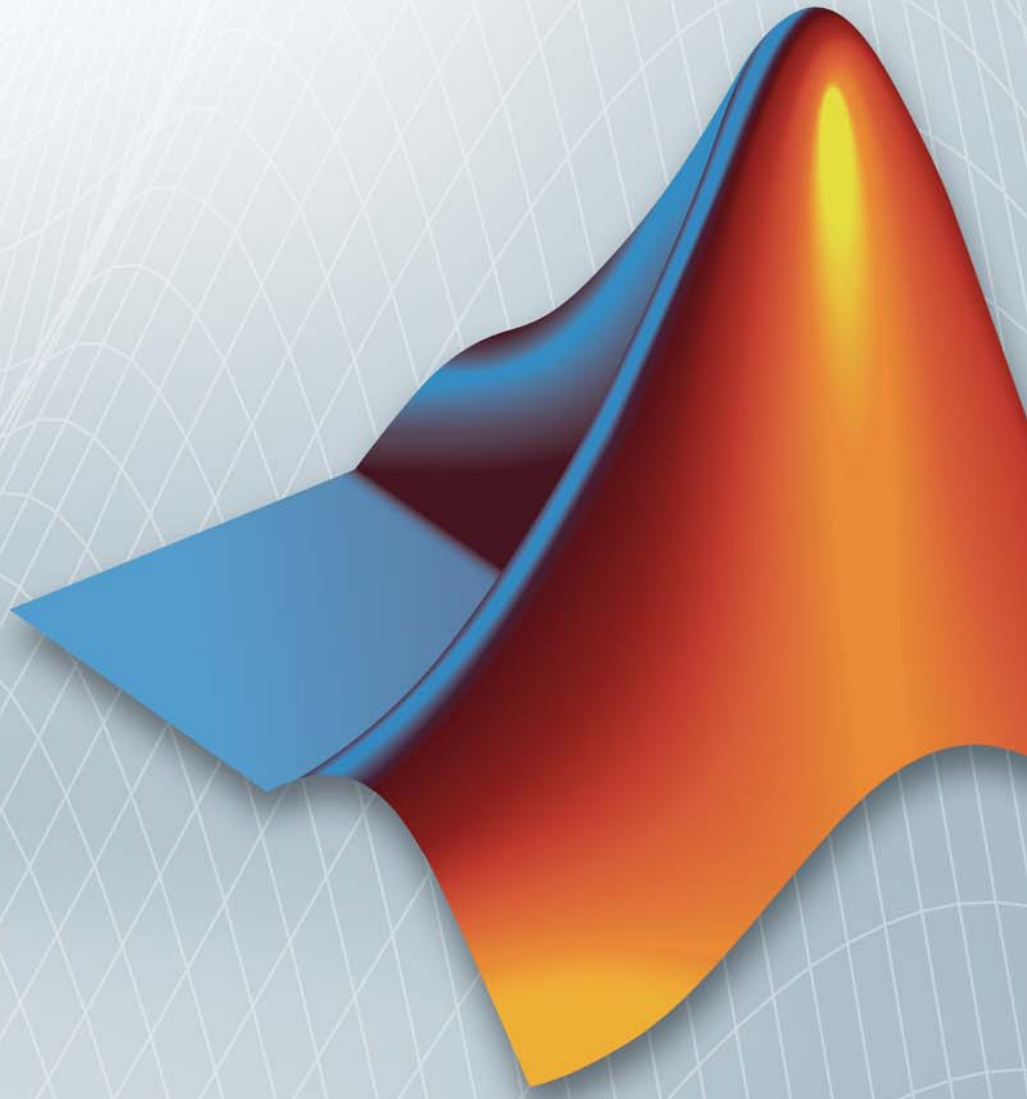
Polyspace® UML Link RH 5

User's Guide



MathWorks®

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Polyspace® UML Link™ RH User's Guide

Revision History

March 2009	Online Only	Revised for Version 5.3 (Release 2009a)
September 2009	Online Only	Revised for Version 5.4 (Release 2009b)
March 2010	Online Only	Revised for Version 5.5 (Release 2010a)
September 2010	Online Only	Revised for Version 5.6 (Release 2010b)
April 2011	Online Only	Revised for Version 5.7 (Release 2011a)

Contents

Polyspace UML Link RH User's Guide

1

1

Polyspace UML Link RH User's Guide

Basic Workflow

In a collaborative Model-Driven Development (MDD) environment, software run-time errors can be produced by either design issues in the model or faulty handwritten code. You may be able to detect the flaws using code reviews and intensive testing. However, these techniques are time-consuming and expensive.

Through Polyspace® UML Link™ RH software, you can apply Polyspace verification to C, C++ and Ada code that you generate from your IBM® Rational® Rhapsody® model. As a result, you can detect run-time errors and automatically identify model flaws quickly and early during the design process.

For information about installing and using IBM Rational Rhapsody, go to www-01.ibm.com/software/awdtools/rhapsody/.

The workflow for using the Polyspace UML Link RH add-in within the IBM Rational Rhapsody MDD environment is:

- Install the add-in. See "Installing Polyspace UML Link RH " in the *Polyspace Installation Guide*.
- Integrate the Polyspace add-in with your Rhapsody project. See "Integrating and Upgrading the Polyspace Add-In" on page 1-6.
- If required, specify Polyspace configuration options in the Polyspace Launcher. See "Configuring Verification Options" on page 1-9.
- Specify the `include` path to your operating system (environment) header files and run verification. See "Running a Verification" on page 1-10 and "Monitoring a Verification" on page 1-14.
- View results, analyze errors, and locate faulty code within model. See "Viewing Polyspace Results" on page 1-15 and "Locating Faulty Code in Rhapsody Model" on page 1-16.

Accessing Polyspace Functions

The Polyspace Panel has various button options that allow you to set up and perform Polyspace functions.

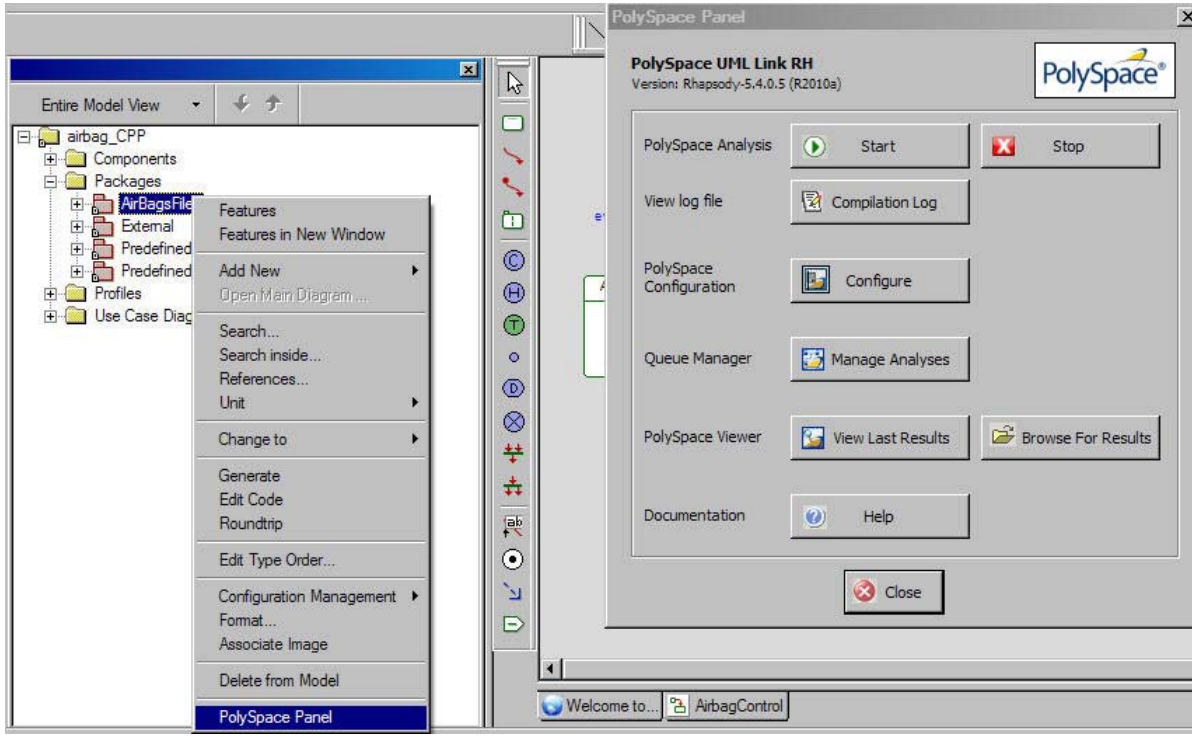
To open the Polyspace Panel in the Rhapsody editor:

1. In the **Entire Model View**, open the model that you want to verify.

   For example, `airbag_CPP.rpy` in *`PolyspaceInstallCommon`*`/PolySpaceUMLLink/example`. *`PolyspaceInstallCommon`* is the installation location of the Polyspace common folder.

2. Select files or packages, for example, **AirBagFiles** under **Packages**.

3. Right-click **AirBagFiles**. From the context menu, select **PolySpace Panel**. The Polyspace Panel opens.



On the Polyspace Panel, you see the following buttons:

- **Start** — Start verification. See "Running a Verification" on page 1-10.
- **Stop** — Stop client-based verification. See "Running a Verification" on page 1-10.
- **Compilation Log** — View latest compilation log. See "Monitoring a Verification" on page 1-14.
- **Configure** — Specify verification options. See "Configuring Verification Options" on page 1-9.
- **Manage Analyses** — Open Polyspace Queue Manager. See "Monitoring a Verification" on page 1-14.
- **View Last Results** — View results of last verification. See "Viewing Polyspace Results" on page 1-15.

- **Browse For Results** — Browse or open verification results. See "Viewing Polyspace Results" on page 1-15.
- **Help** — Open PDF help document.
- **Close** — Close Polyspace Panel.

Integrating and Upgrading the Polyspace Add-In

In this section...

Integrating the Polyspace Add-In with a Rhapsody Project

The Polyspace add-in is written using the Visual Basic® extension provided by Rhapsody. This add-in installs itself in new Rhapsody projects by using the copyVBA feature in the `rhapsody.ini` file.

Note For Rhapsody versions until 7.3, the environment variable `%WINDIR%` specifies the location of `rhapsody.ini`. With later versions, this file is located in the parent Rhapsody installation folder.

For Rhapsody projects that already exist, you must copy the Polyspace Visual Basic add-in from `<PolyspaceCommonInstall>\PolySpaceUMLLink\bin\polyspace.vba` to the project folder. Then, rename the add-in *`project_name`*`.vba`, which replaces the existing VBA file. *`PolyspaceInstallCommon`* is the installation location of the Polyspace common folder.

If you use Rhapsody Visual Basic, you carry out the integration as follows:

1. Using the Rhapsody Visual Basic editor, export each form and module for the existing code.
2. Close your model.
3. Copy the `polyspace.vba` file into the model folder and rename the file *`project_name`*`.vba`.
4. Reopen the model, and use the Visual Basic editor to import each form and module that was exported in step 1.

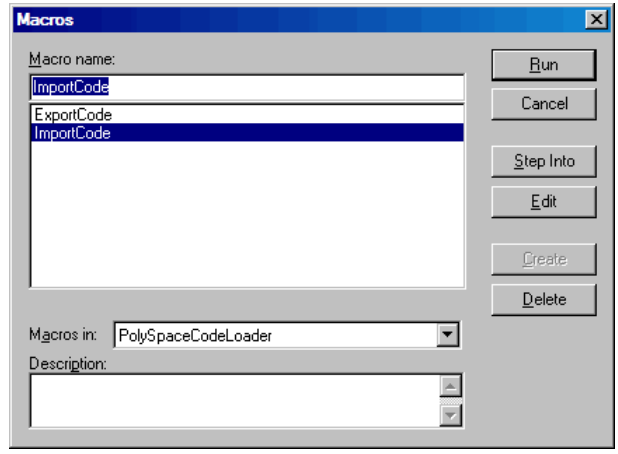
You can also update the master `polyspace.vba` file with the contents of *`project_name`*`.vba` for use in new projects.

Upgrading Polyspace Software

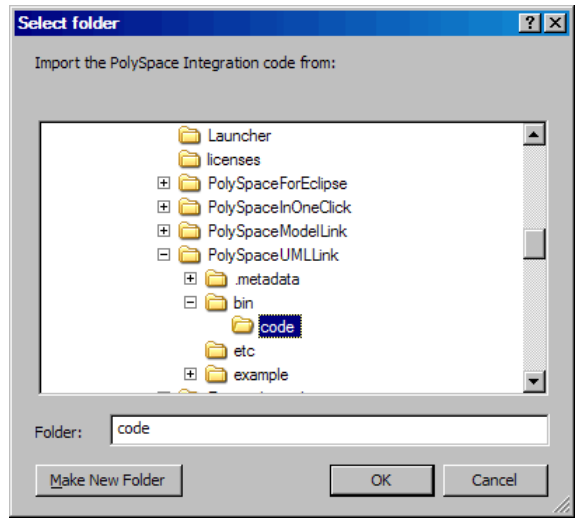
When you upgrade your Polyspace software, the VBA code in existing Rhapsody projects is not automatically updated.

To update your Polyspace integration:

1. Open your Rhapsody model.
2. Select **Tools > VBA > Macros**.



3. In the Macros dialog box, select **ImportCode**.
4. Click **Run**.



5 In the Select folder dialog box, select the folder containing the Polyspace UML Link RH code. By default, this code is located in `Polyspace_Common/PolySpaceUMLLink/bin/code`.

6 Click **OK**.

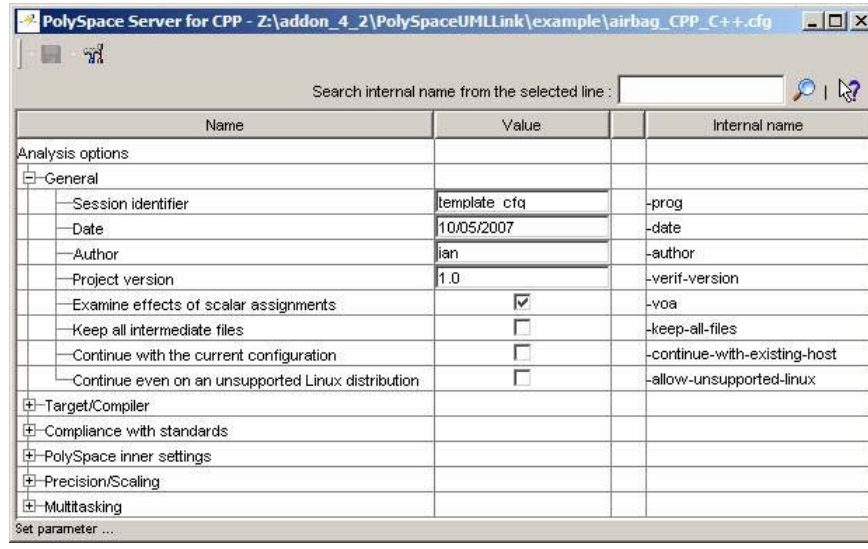
The software imports the appropriate code modules from the selected folder.

Note You can also use this VBA macro to export Polyspace code from your model. You may want to do this when merging VBA code from two separate projects. When exporting code from a model, be careful not to overwrite the code in the `PolySpaceUMLLink` folder.

Configuring Verification Options

To specify options for your verification:

1. Select your files and open the Polyspace Panel. See "Accessing Polyspace Functions" on page 1-3.
2. Click **Configure** to open a scaled-down version of the Polyspace Launcher.



3. Select options for your verification.
4. To save your options, in the top left corner, click the disk button.

For information on how to choose your options, see the *Option Description* section in the appropriate *Polyspace Products Reference Guide*.

Running a Verification

To start a verification:

1 On the Polyspace Panel, click **Start**.

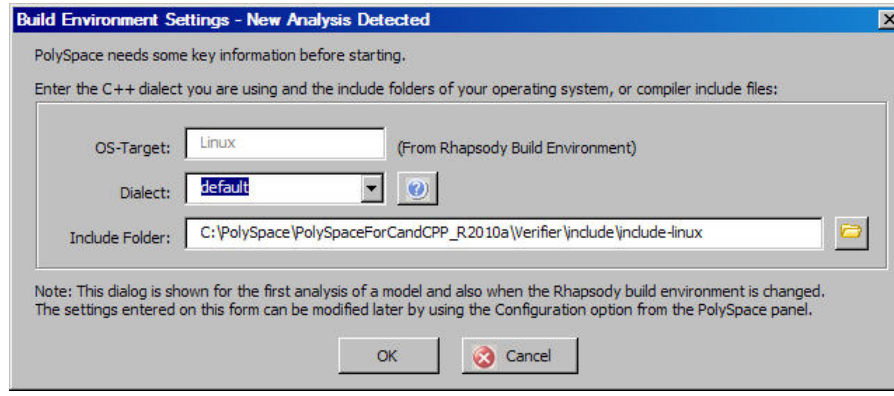
If you are starting the first verification of a model, or if the Rhapsody configuration environment has changed since the last verification, the software opens the Build Environment Settings dialog box.

The software detects the operating system target from the model environment and sets the **OS-Target** field automatically. Specify:

- **Dialect** if required
- **Include Folder** — Path to your operating system (environment) header files

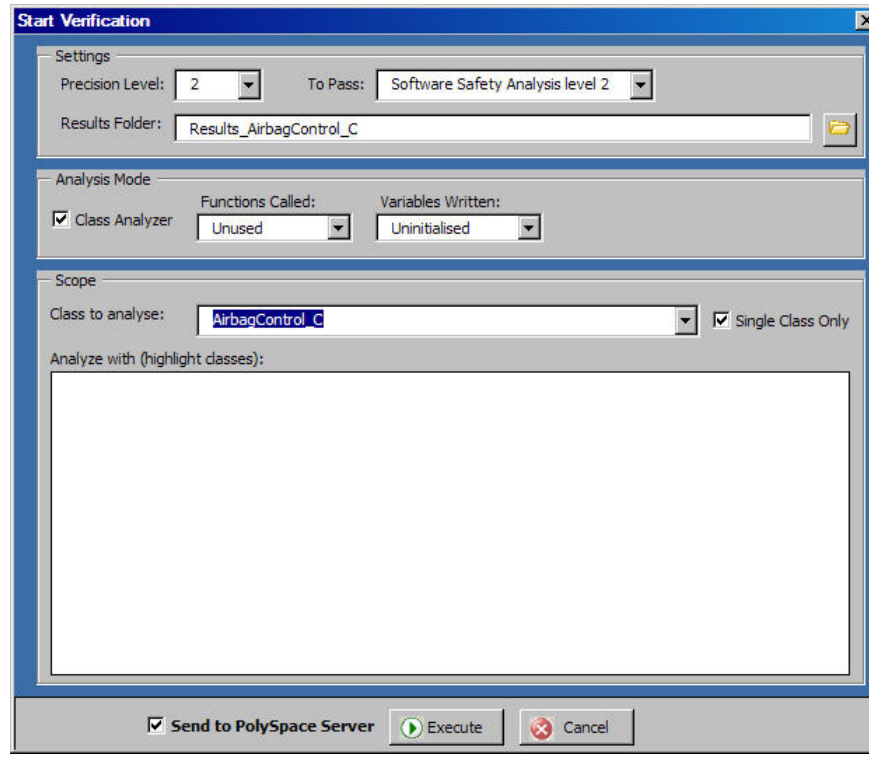
If the software detects Linux® , then the software:

- Sets the **Dialect** field to `default`
- Specifies the path to the Linux header files supplied with Polyspace software in the **Include Folder** field.



Note Before starting a verification, make sure that the generated code for the model is up to date.

2 Click **OK**.



3 In the Start Verification dialog box, under **Settings**, specify values for the following fields:

- **Precision Level** — Precision of your verification
- **To Pass** — Number of passes that you want the software to carry out
- **Results Folder** — Location of the results folder

Note The software sets the **Results Folder** field automatically when you specify or change:

- The type of verification that you want to perform
- The package that you want to verify

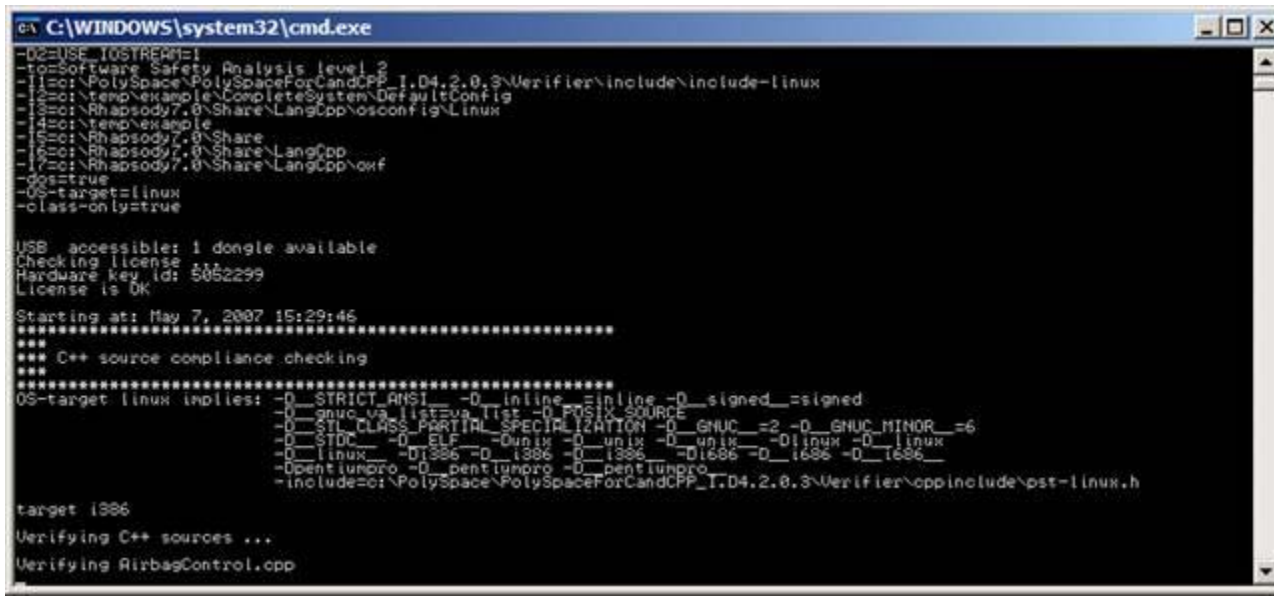
After you select your verification type and package, you can alter the contents of the **Results Folder** field.

4 Under **Analysis Mode**, select the type of verification that you want to perform. For more information, see the *Option Description* section in the appropriate *Polyspace Products Reference Guide*.

5 Under **Scope**, select the file, class, or package that you want to verify.

6 If you want to send your verification to the Polyspace Server, select the **Remote Mode** check box. If you want to perform the verification locally, clear this check box.

7 Click **Execute**. A command window opens, showing the phases of the verification that are performed locally.



Note You can change the settings (size of window, number of lines of history, font, and so on) for the command window. Right-click the window title and select **Properties** to open the Properties dialog box. The options available are similar to those available with the Windows® operating system Command Window.

If your verification is client-based, you can stop your verification. On the Polyspace Panel, click **Stop**.

To stop a verification on the Polyspace Server, use the Polyspace Queue Manager. See “Monitoring a Verification” on page 1-14.

Monitoring a Verification

To view the latest compilation log, on the Polyspace Panel, click **Compilation Log**.

If your verification is client-based, you can observe progress in the Command Window that opens when you run the verification.

If your verification is running on a Polyspace Server, on the Polyspace panel, click **Manage Analyses** to display the Polyspace Queue Manager (or Spooler). Use the Polyspace Queue Manager to manage jobs running on any Polyspace Server.

For more information, see:

- "Managing Verification Jobs Using Polyspace Queue Manager" in the *Polyspace Products for Ada User's Guide*
- "Managing Verification Jobs Using the Polyspace Queue Manager" in the *Polyspace Products for C User's Guide*

Viewing Polyspace Results

To view the results from the last completed verification, on the Polyspace Panel, click **View Last Results**.

If no results are available (that is, the results are still on the server), the software prompts you to open the Polyspace Queue Manager. You can download the results using the Polyspace Queue Manager.

To browse through or open all verification results for a given model, on the Polyspace Panel, click **Browse for Results** .

For more information on Polyspace verification results, see:

- "Reviewing Verification Results" in the *Polyspace Products for Ada User's Guide*
- "Reviewing Verification Results" in the *Polyspace Products for C User's Guide*

Declarations for C Functions Without Arguments

By default, Rhapsody generates declarations for functions without any parameters, using the form:

```
void my_function()
```

rather than:

```
void my_function(void)
```

This can result in the following Polyspace compilation error:

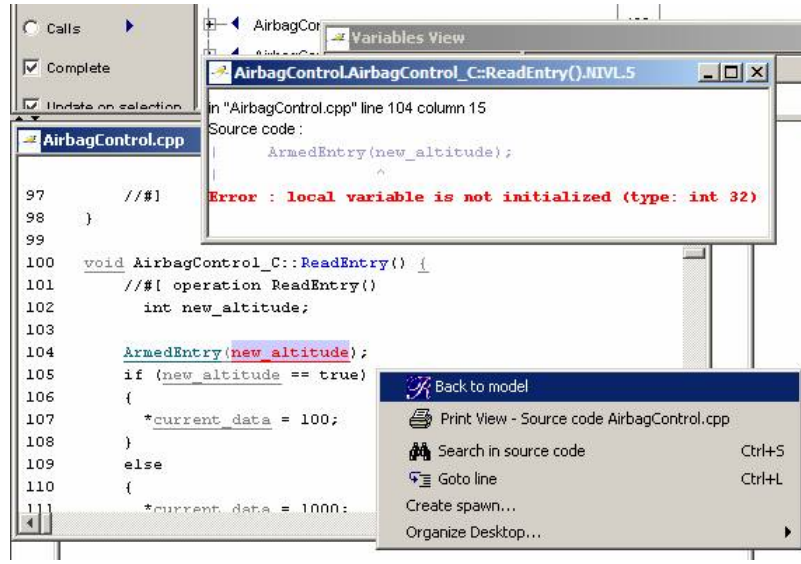
```
Fatal error: function 'my_function' has unknown prototype.
```

To avoid this problem, in Rhapsody, at the project level, set the property `C_CG::Configuration::EmptyArgumentListName` to `void`.

Locating Faulty Code in Rhapsody Model

To identify the faulty code within your Rhapsody model using Polyspace verification results:

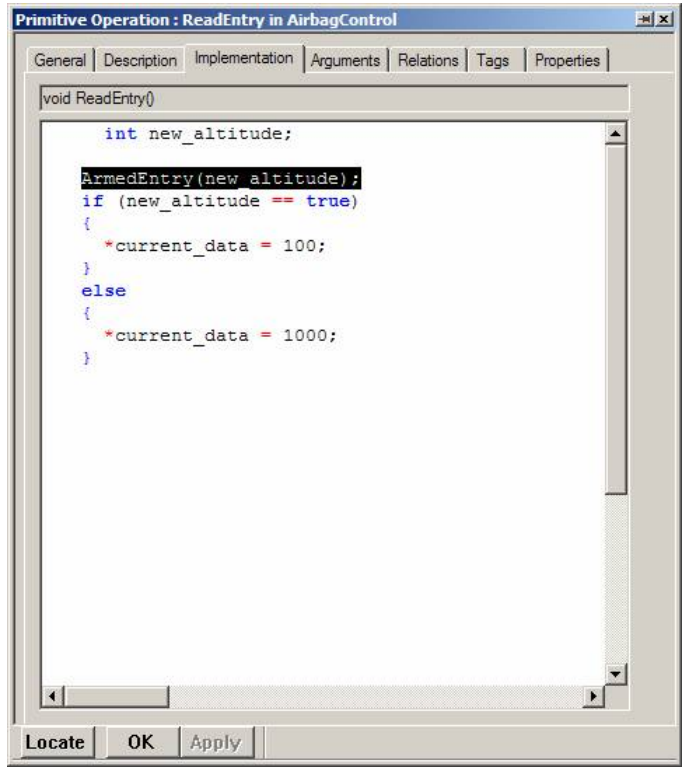
1. In the Polyspace Viewer, navigate to an error, for example, a non-initialized variable at line 104 of `Airbag Control_C`.
2. Right-click the error. From the context menu, select **Back to Model**.



Note For the **Back To Model** command to work, you must have your Rhapsody model open.

The **Back To Model** command is available only if the Polyspace check is enclosed by the tags `//#[` and `]#//`.

The software locates the faulty code within your Rhapsody model. Depending on the Rhapsody configuration, the faulty code appears either in a dialog box or in the code view.



Limitations

You can use the **Back to Model** command only if the source code lines that contain a Polyspace check do not also contain a macro. Otherwise, the software produces the following warning:

```
Unable to go back to the UML model from this location
```

For locations where Rhapsody does not support the **Back to Model** command, or for code that does not belong to the model, the software produces the following warning when you try to use the **Back to Model** command:

```
No element found in model
```

Template Configuration Files

In this section...

Using Template Configuration Files

The first time you perform a verification, the software copies a template, Polyspace configuration file, from *PolyspaceInstallCommon*/PolySpaceForUML/etc/template_*language*.cfg to the project folder and renames this file:

model_language.cfg

where

- *model* is the name of your model
- *language* is the name of the language that the model targets, that is, C, C++, or Ada

You can update the template .cfg file by one of the following means:

- Editing it through the Polyspace Launcher
- Double-clicking the file in a Windows Explorer window
- Replacing the template file with a copy of the .cfg file from a Rhapsody model folder

You can then share a configuration among project members and use the configuration with other projects.

Default Configuration Options

In the template_C++.cfg file, the following are the default option values:

```
-lang=CPP
-prog=template_cfg
```

```
-results-dir=r->results
-allow-undef-variables=true
-respect-types-in-globals=true
-respect-types-in-fields=true
-dos=true
-target=i386
-D=[OM_NO_FRAMEWORK_MEMORY_MANAGER]
-to=7
-OS-target=no-predefined-OS
```

In the `template_Ada.cfg` file, the following are the default options:

```
-lang=ADA
-prog=template_cfg
-results-dir=r->results
-allow-undef-variables=true
-respect-types-in-globals=true
-respect-types-in-fields=true
-dos=true
-target=i386
-D=[OM_NO_FRAMEWORK_MEMORY_MANAGER]
-to=7
-OS-target=no-predefined-OS
```