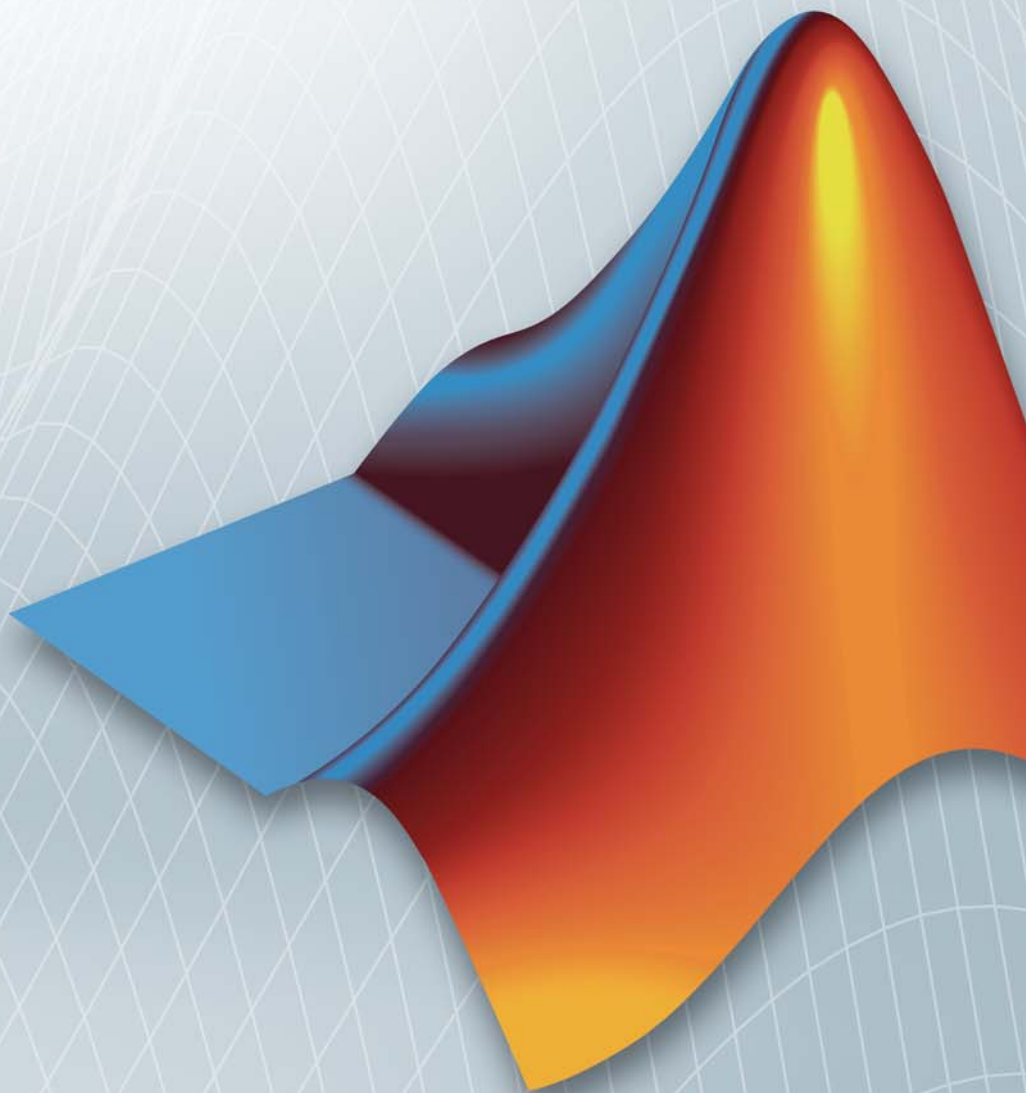


Polyspace® UML Link RH

User's Guide

R2012b



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

© COPYRIGHT 1999–2012 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

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)
September 2011	Online Only	Revised for Version 5.8 (Release 2011b)
March 2012	Online Only	Revised for Version 5.9 (Release 2012a)
September 2012	Online Only	Revised for Version 5.10 (Release 2012b)

Polyspace UML Link RH User's Guide

1

Product Description	1-2
Code Verification Approach	1-3
Accessing Polyspace Features	1-4
Adding Polyspace Profile to Model	1-7
Configuring Verification Options	1-9
Running a Verification	1-12
Monitoring a Verification	1-14
Viewing Polyspace Results	1-15
Declarations for C Functions Without Arguments	1-15
Locating Faulty Code in Rhapsody Model	1-16
Template Configuration Files	1-18
Using Template Configuration Files	1-18
Default Configuration Options	1-18

Polyspace UML Link RH User's Guide

- “Product Description” on page 1-2
- “Code Verification Approach” on page 1-3
- “Accessing Polyspace Features” on page 1-4
- “Adding Polyspace Profile to Model” on page 1-7
- “Configuring Verification Options” on page 1-9
- “Running a Verification” on page 1-12
- “Monitoring a Verification” on page 1-14
- “Viewing Polyspace Results” on page 1-15
- “Locating Faulty Code in Rhapsody Model” on page 1-16
- “Template Configuration Files” on page 1-18

Product Description

Polyspace® UML Link™ RH software extends Polyspace Client™ for C/C++ and Polyspace Server™ for C/C++ products with tools that let you trace Polyspace results from generated C++ code directly to your IBM® Rational® Rhapsody® models. As a result, you can identify parts of the model that do not generate code with run-time errors, and fix design problems that cause run-time errors. You can verify a mix of generated and hand-written code before it is compiled.

Code Verification Approach

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/.

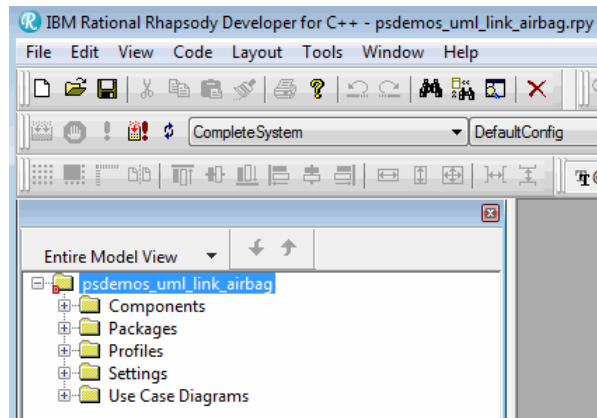
The approach for using the Polyspace UML Link RH add-in within the IBM Rational Rhapsody MDD environment is:

- Install the add-in. See “Getting Started with Polyspace UML Link RH”.
- Integrate the Polyspace add-in with your Rhapsody project. See “Adding Polyspace Profile to Model” on page 1-7.
- If required, specify Polyspace configuration options in the Polyspace verification environment. 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 1-12 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.

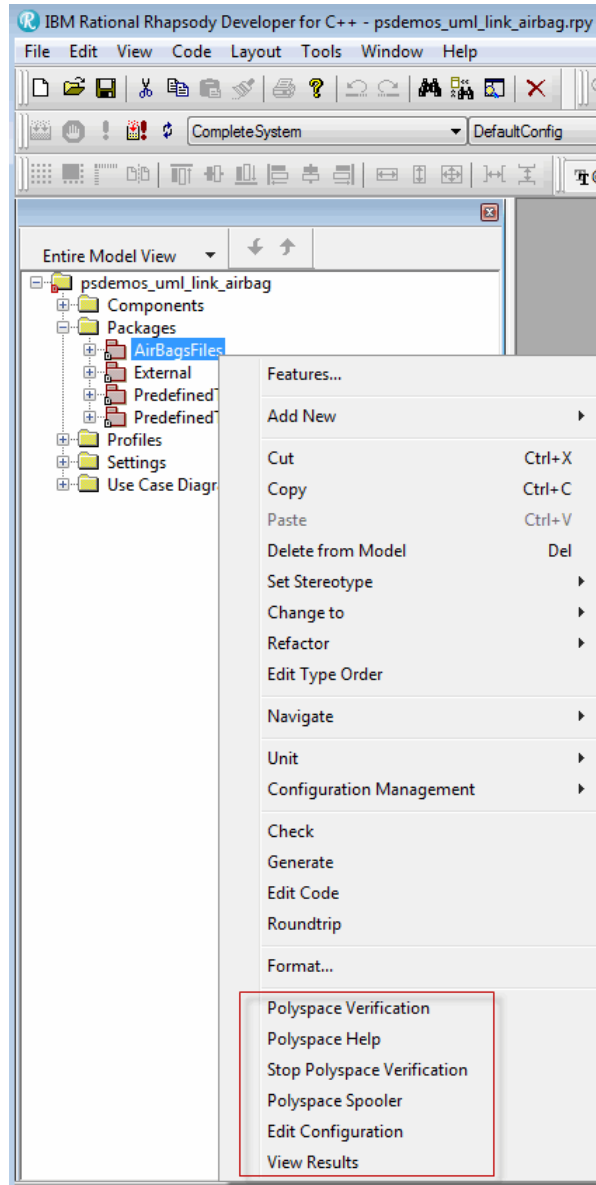
Accessing Polyspace Features

To access Polyspace features in the Rhapsody editor:

- 1 Open the model that you want to verify. For example, `psdemos_uml_link_airbag.rpy` in `Polyspace_Common/PolyspaceUMLLink/psdemos`. *Polyspace_Common* is the installation location of the Polyspace common folder.



- 2 In the **Entire Model View**, expand the Packages node.
- 3 Right-click a package, for example, **AirBagFiles**.



You see the following Polyspace functions in the context menu:

- **Polyspace Verification** — Start verification. See “Running a Verification” on page 1-12.
- **Polyspace Help** — Open help.
- **Stop Polyspace Verification** — Stop client-based verification. See “Running a Verification” on page 1-12.
- **Polyspace Spooler** — Open Polyspace Queue Manager. See “Monitoring a Verification” on page 1-14.
- **Edit Configuration** — Specify verification options. See “Configuring Verification Options” on page 1-9.
- **View Results** — View verification results. See “Viewing Polyspace Results” on page 1-15.

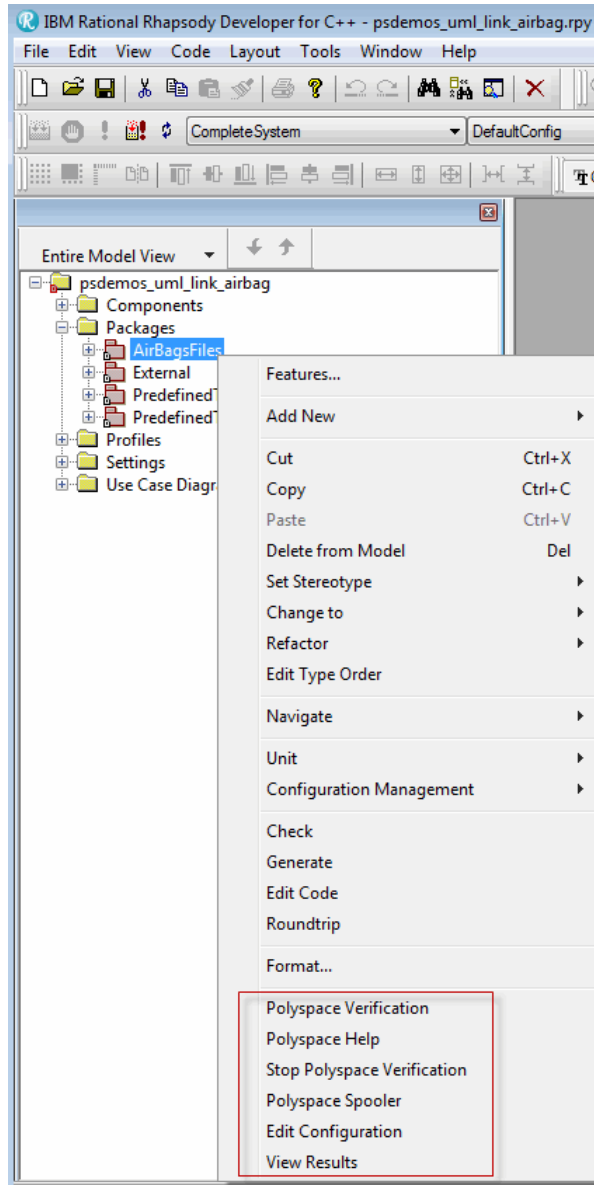
Note You must add the Polyspace profile to your model before you try to access Polyspace functions. See “Adding Polyspace Profile to Model” on page 1-7.

Adding Polyspace Profile to Model

You must add the Polyspace profile to your model before you try to access Polyspace functions:

- 1** In the Rhapsody editor, select **File > Add Profile to Model**. The Add Profile to Model dialog box opens.
- 2** Navigate to the folder *Polyspace_Common/PolyspaceUMLLink/profiles/Polyspace*.
- 3** Select the file *Polyspace.sbs*. Then click **Open**.

Now, if you right-click a package or file, you see Polyspace functions in the context menu.

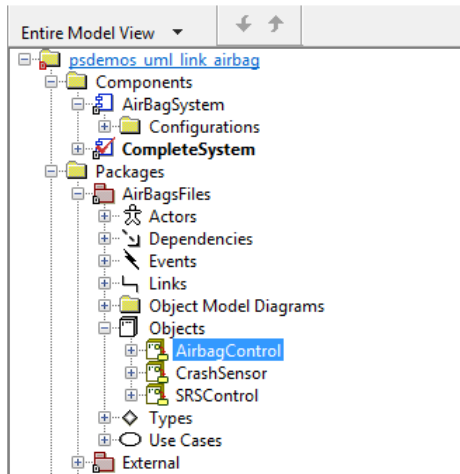


Polyspace Verification is also available from the **Tools** menu.

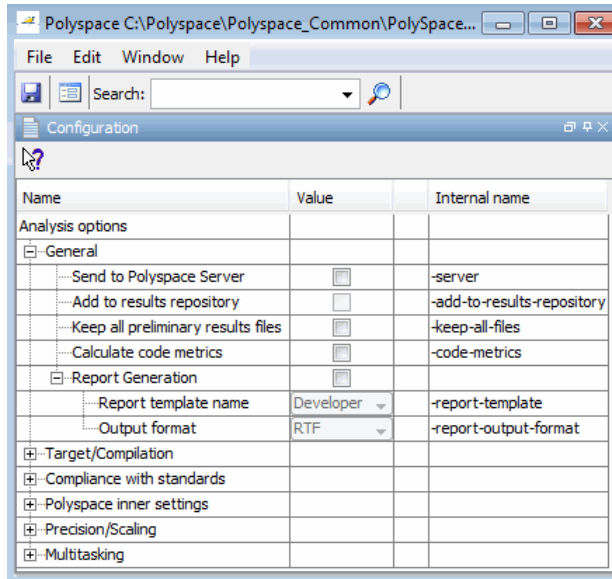
Configuring Verification Options

To specify options for your verification:

- 1 In the **Entire Model View**, right-click a package or class, for example, AirbagControl.



- 2 From the context menu, select **Edit Configuration**. The **Configuration** pane of the Polyspace verification environment opens.



3 Select options for your verification. In particular, you must specify the following:

- **Target operating system** (-OS-target)
- **Dialect** (-dialect)
- **Include Folders** (-I) — Path to your operating system (environment) header files.

Note Polyspace provides Linux® header files for C++ in include-gnu, which is located within the Polyspace product folder. For example,

```
C:\Polyspace\PolyspaceForCandCPP_R2011b\Verifier\include\include-gnu
```

4 To save your options, in the top left corner, click the disk button.

For information on how to choose your options, see:

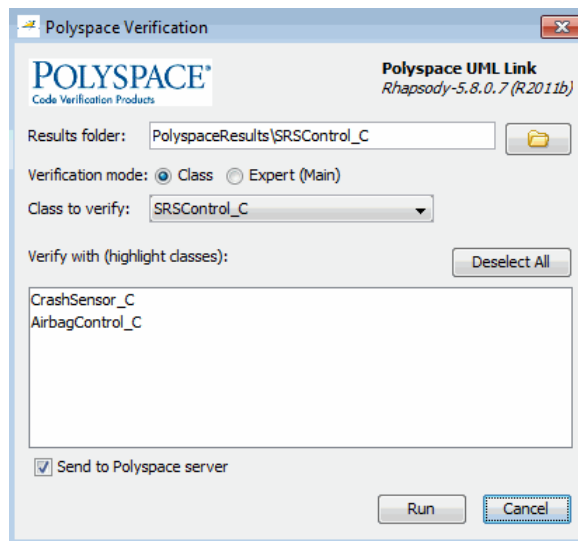
- “Analysis Option Reference” for Ada

- “Analysis Options for C Code”
- “Analysis Options for C++ Code”

Running a Verification

To start a verification:

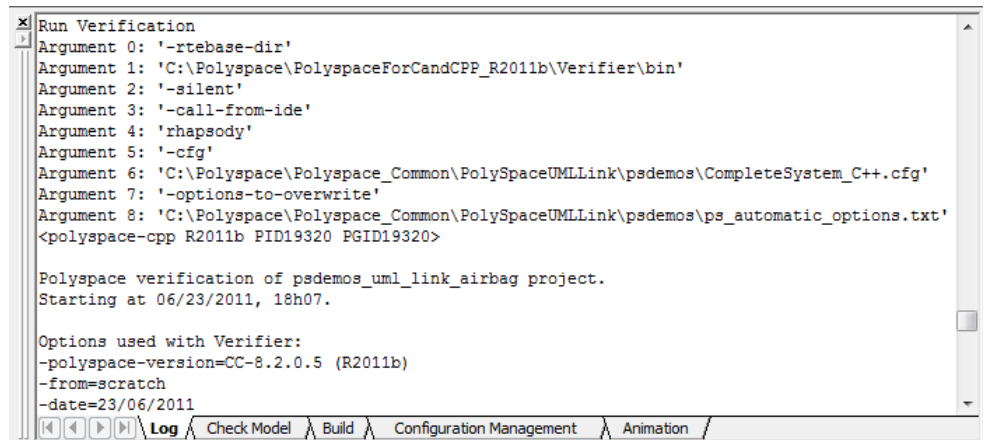
- 1 In the Rhapsody editor, select **Tools > Polyspace Verification**. The software opens the Polyspace Verification dialog box.



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

- 2 In the Results folder field, specify a location for your verification results.
- 3 Select the **Verification mode**:
 - **Class** — Select a specific class from the **Class to verify** drop-down list. In addition, under **Verify with (highlight classes)**, you can select other classes from the displayed list, for example, `CrashSensor_C`.

- **Expert** — The software verifies code according to the **Generate a main** (-main-generator) options that you specify.
- 4 If you want to run the verification on your Polyspace server, select **Send to Polyspace server**.
 - 5 Click **Run**. You see verification messages on the **Log** tab of the Rhapsody editor.



```

Run Verification
Argument 0: '-rtebase-dir'
Argument 1: 'C:\Polyspace\PolyspaceForCandCPP_R2011b\Verifier\bin'
Argument 2: '-silent'
Argument 3: '-call-from-ide'
Argument 4: 'rhapsody'
Argument 5: '-cfg'
Argument 6: 'C:\Polyspace\Polyspace_Common\PolySpaceUMLLink\psdemos\CompleteSystem_C++.cfg'
Argument 7: '-options-to-override'
Argument 8: 'C:\Polyspace\Polyspace_Common\PolySpaceUMLLink\psdemos\ps_automatic_options.txt'
<polyspace-cpp R2011b PID19320 PGID19320>

Polyspace verification of psdemos_uml_link_airbag project.
Starting at 06/23/2011, 18h07.

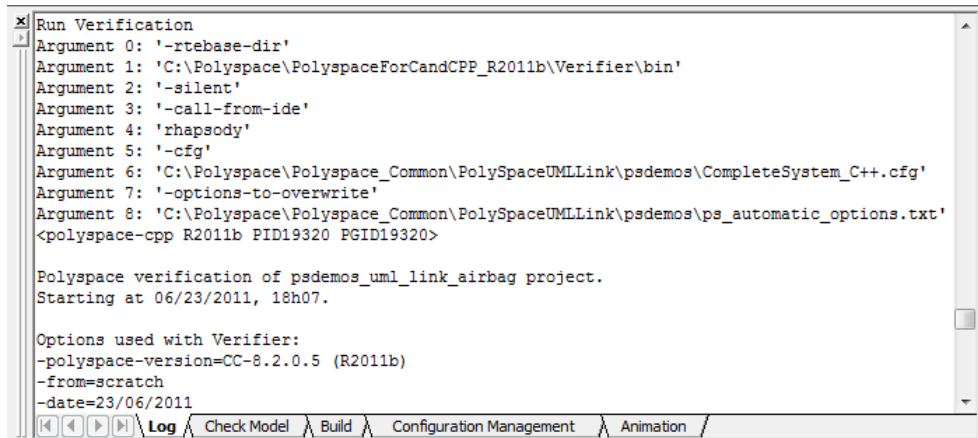
Options used with Verifier:
-polyspace-version=CC-8.2.0.5 (R2011b)
-from=scratch
-date=23/06/2011
  
```

If your verification is client-based, you can stop your verification. In the **Entire Model View**, right-click, for example, a package or a class. From the context menu, select **Stop Polyspace Verification**.

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

Monitoring a Verification

If your verification is client-based, you can observe progress on the **Log** tab of the Rhapsody editor.



```

Run Verification
Argument 0: '-rtebase-dir'
Argument 1: 'C:\Polyspace\PolyspaceForCandCPP_R2011b\Verifier\bin'
Argument 2: '-silent'
Argument 3: '-call-from-ide'
Argument 4: 'rhapsody'
Argument 5: '-cfg'
Argument 6: 'C:\Polyspace\Polyspace_Common\PolySpaceUMLLink\psdemos\CompleteSystem_C++.cfg'
Argument 7: '-options-to-overwrite'
Argument 8: 'C:\Polyspace\Polyspace_Common\PolySpaceUMLLink\psdemos\ps_automatic_options.txt'
<polyspace-cpp R2011b PID19320 PGID19320>

Polyspace verification of psdemos_uml_link_airbag project.
Starting at 06/23/2011, 18h07.

Options used with Verifier:
-polyspace-version=CC-8.2.0.5 (R2011b)
-from=scratch
-date=23/06/2011
  
```

Log | Check Model | Build | Configuration Management | Animation

If your verification is running on a Polyspace Server, in the **Entire Model View**, right-click, for example, a package or a class. From the context menu, select **Polyspace Spooler** 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:

- “Verification Job Management” for Ada
- “Verification Job Management” for C/C++

Viewing Polyspace Results

To view results from the last completed verification, in the **Entire Model View**, right-click, for example, a package or a class. From the context menu, select **View Results**. The Polyspace verification environment opens, displaying results in the Results Manager perspective.

For more information on Polyspace verification results, see:

- “Run-Time Error Review” for Ada
- “Run-Time Error Review” for C/C++

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:

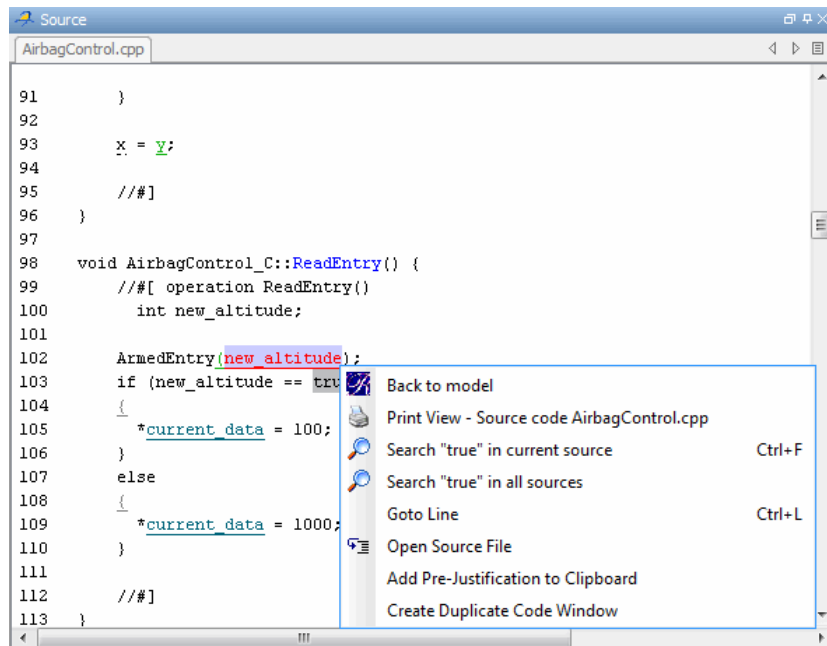
```
Fatal error: function 'my_function' has unknown prototype.
```

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 Results Manager perspective of the Polyspace verification environment, navigate to an error, for example, a non-initialized variable at line 102 of Airbag Control_C.
- 2 In the Source pane, right-click the error. From the context menu, select **Back to model**.



The screenshot shows a source code editor window titled "Source" with a tab for "AirbagControl.cpp". The code is as follows:

```
91     }
92
93     x = y;
94
95     /*#]
96   }
97
98   void AirbagControl_C::ReadEntry() {
99     /*[ operation ReadEntry()
100       int new_altitude;
101
102     ArmedEntry(new_altitude);
103     if (new_altitude == true)
104     {
105       *current_data = 100;
106     }
107     else
108     {
109       *current_data = 1000;
110     }
111
112     /*#]
113 }
```

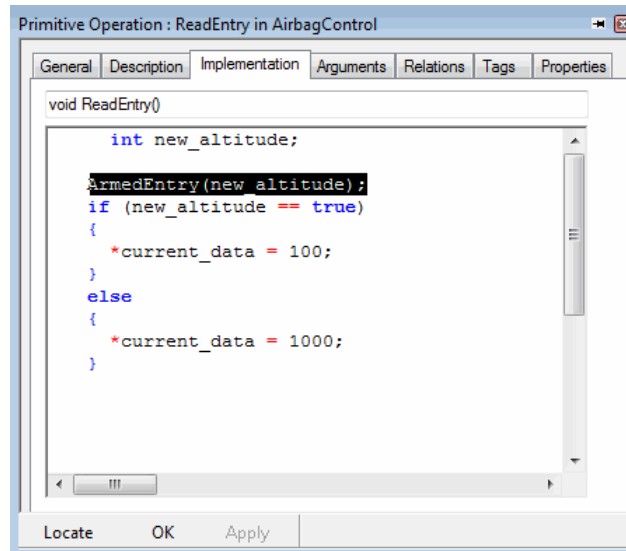
A context menu is open over line 102, with the following options:

- Back to model
- Print View - Source code AirbagControl.cpp
- Search "true" in current source (Ctrl+F)
- Search "true" in all sources
- Goto Line (Ctrl+L)
- Open Source File
- Add Pre-Justification to Clipboard
- Create Duplicate Code Window

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

The **Back To Model** command works best when 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.



The screenshot shows a dialog box titled "Primitive Operation : ReadEntry in AirbagControl". It has a tabbed interface with "Implementation" selected. The code view shows the implementation of the ReadEntry() function. The line `ArmedEntry(new_altitude);` is highlighted in black, indicating it is the faulty code. The code is as follows:

```
void ReadEntry()
{
    int new_altitude;
    ArmedEntry(new_altitude);
    if (new_altitude == true)
    {
        *current_data = 100;
    }
    else
    {
        *current_data = 1000;
    }
}
```

At the bottom of the dialog box, there are buttons for "Locate", "OK", and "Apply".

Template Configuration Files

In this section...

“Using Template Configuration Files” on page 1-18

“Default Configuration Options” on page 1-18

Using Template Configuration Files

The first time you perform a verification, the software copies a template, Polyspace configuration file, from *Polyspace_Common/PolyspaceUMLLink/etc/template_language.cfg* to the project folder. The software also renames the copy *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 verification environment
- 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

The *template_language.cfg* XML files specify the default option values for code verification.

The file *template_Ada.cfg* is:

```
<?xml version="1.0" encoding="UTF-8"?>
<polyspace_project name="template_cfg" language="Ada 95" author="polyspace"
```



```

version="1.0" date="08/04/2011" path="file:/C:/Polyspace/Polyspace_Common
/Rhapsody/PolyspaceUMLLink/etc/template_ADA.cfg">
  <source>
</source>
  <include>
</include>
  <module name="Verification_1" isactive="true">
    <source>
</source>
    <optionset name="template_cfg" isactive="true">
      <option flagname="-OS-target">no-predefined-OS</option>
</optionset>
  </module>
</polyspace_project>

```

The file `template_C.cfg` is:

```

<?xml version="1.0" encoding="UTF-8"?>
<polyspace_project name="template_cfg" language="C" author="polyspace"
version="1.0" date="08/04/2011" path="file:/C:/Polyspace/Polyspace_Common
/Rhapsody/PolyspaceUMLLink/etc/template_C.cfg">
  <source>
</source>
  <include>
</include>
  <module name="Verification_1" isactive="true">
    <source>
</source>
    <optionset name="template_cfg" isactive="true">
      <option flagname="-OS-target">no-predefined-OS</option>
      <option flagname="-allow-undef-variables">true</option>
      <option flagname="-respect-types-in-fields">true</option>
      <option flagname="-respect-types-in-globals">true</option>
    </optionset>
  </module>
</polyspace_project>

```

The file `template_C++.cfg` is:

```

<?xml version="1.0" encoding="UTF-8"?>
<polyspace_project name="template_cfg" language="C++" author="polyspace"

```

```
version="1.0" date="08/04/2011" path="file:/C:/Polyspace/Polyspace_Common
/Rhapsody/PolyspaceUMLLink/etc/template_C++.cfg">
  <source>
</source>
  <include>
</include>
  <module name="Verification_1" isactive="true">
    <source>
</source>
    <optionset name="template_cfg" isactive="true">
      <option flagname="-D">[OM_NO_FRAMEWORK_MEMORY_MANAGER]</option>
      <option flagname="-OS-target">no-predefined-OS</option>
      <option flagname="-allow-undef-variables">true</option>
      <option flagname="-dialect">gnu</option>
      <option flagname="-respect-types-in-fields">true</option>
      <option flagname="-respect-types-in-globals">true</option>
      <option flagname="-target">i386</option>
    </optionset>
  </module>
</polyspace_project>
```