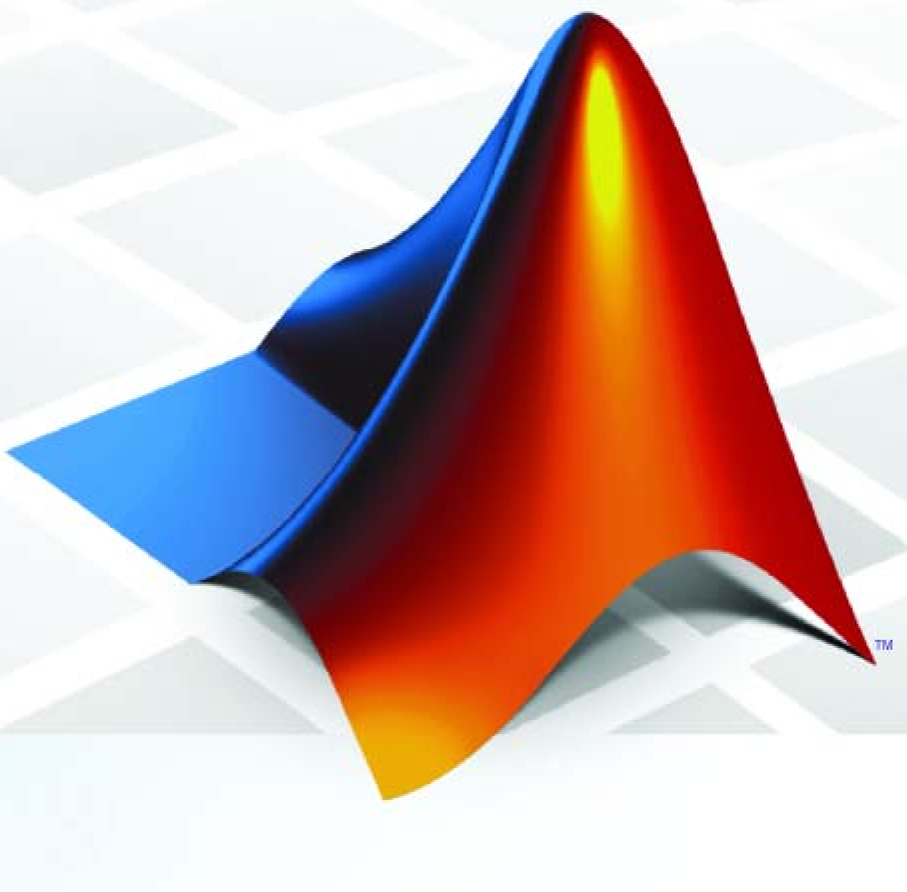
PolySpace® Products for Ada 5
Getting Started Guide



The MathWorks™
Accelerating the pace of engineering and science

Revision History

March 2008	First printing	Revised for Version 5.1 (Release 2008a)
October 2008	Second printing	Revised for Version 5.2 (Release 2008b)
March 2009	Third printing	Revised for Version 5.3 (Release 2009a)

Contents

1 Introduction to PolySpace Products for Verifying Ada Code

2 Setting Up a Project File

4 Reviewing Verification Results

Index

Introduction to PolySpace Products for Verifying Ada Code

Product Overview

In this section...

Ensures Software Reliability

You can ensure the reliability of your Ada applications by using PolySpace® verification software to prove code correctness and identify run-time errors. Using advanced verification techniques, PolySpace software performs an exhaustive verification of your source code.

Because PolySpace software verifies all possible executions of your code, it can identify code that:

- Never has an error
- Always has an error
- Is unreachable
- Might have an error

With this information, you can be confident that you know how much of your code is run-time error free, and you can improve the reliability of your code by fixing the errors.

Decreases Development Time

Using PolySpace verification software reduces development time by automating the verification process and helping you to efficiently review verification results. You can use it at any point in the development process, but using it during early coding phases allows you to find errors when it is less costly to fix them.

You use PolySpace software to verify Ada source code prior to compilation. To verify the source code, you set up verification parameters in a project, run the verification, and review the results. This process takes significantly less time than using manual methods or using tools that require you to modify code or run test cases.

A graphical user interface helps you to efficiently review verification results. Results are color-coded:

- Green indicates code that never has an error.
- Red indicates code that always has an error.
- Gray indicates unreachable code (dead code).
- Orange indicates unproven code (code that might have an error).

This color-coding system helps you to identify errors quickly. You will spend less time debugging because you can see the exact location of an error in the source code. After you fix errors, you can easily run the verification again.

Using PolySpace verification software helps you to use your time effectively. Because you know which parts of your code are error-free, you can focus on the code that has definite errors or might have errors.

Reviewing the code that might have errors (orange code) can be time-consuming, but PolySpace software helps you with the review process. You can use filters to focus on certain types of errors or you can allow the software to identify the code that you should review.

Improves the Development Process

PolySpace software makes it easy to share verification parameters and results, allowing the development team to work together to improve product reliability. Once verification parameters have been set up, developers can reuse them for other packages in the same application.

PolySpace verification software supports code verification throughout the development process:

- An individual developer can find and fix run-time errors during the initial coding phase.
- Quality assurance can check overall reliability of an application.

- Managers can monitor application reliability by generating reports from the verification results.

Product Components

The PolySpace products for verifying Ada code are:

PolySpace® Client™ for Ada
PolySpace® Server™ for Ada

The user interface includes:

- The *Launcher* for setting up verification parameters and starting verifications.
- The *Viewer* for reviewing verification results.
- The *Spooler* for managing verifications that run on a server and downloading results from a server to a client.

Installing PolySpace Products

In this section...

Finding the Installation Instructions

The tutorials in this guide require both PolySpace Client for Ada and PolySpace Server for Ada. Instructions for installing PolySpace products are in the *PolySpace Installation Guide*. Before running PolySpace products, you must also obtain and install the necessary licenses.

Obtaining Licenses for PolySpace Client for Ada and PolySpace Server for Ada Products

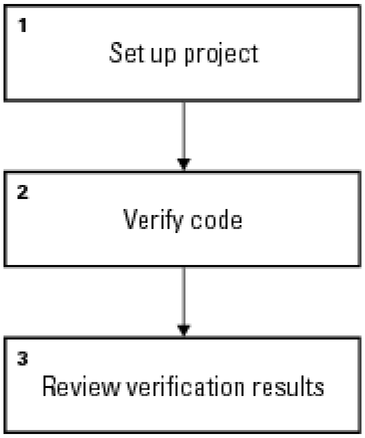
See "PolySpace License Installation" in the *PolySpace Installation Guide* for information about obtaining and installing licenses for PolySpace products.

Working with PolySpace Software

In this section...

Basic Workflow

The basic workflow for using PolySpace software to verify Ada source code is:



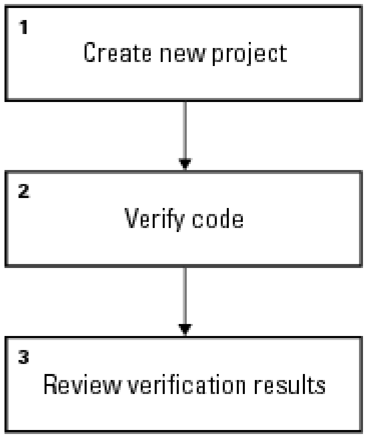
In this workflow, you:

1. Use the Launcher to set up a project file.
2. Verify code on a server or client.

   You can use the Launcher to start the verification or you can select files from a Microsoft® Windows® folder and send them to thePolySpace software for verification. For verifications that run on a server, you can use the Spooler to manage the verifications and download the results to a client.
3. Use the Viewer to review verification results.

The Workflow in This Guide

The tutorials in this guide take you through the basic workflow, including the different options for running verifications. The workflow that you will follow in this guide is:



In this workflow, you will:

1 Create a new project that you can use for the other steps in the workflow.

   This step is in the tutorial Chapter 2, "Setting Up a Project File".

2 Verify a single package using demo Ada source code.

   This step is in the tutorial Chapter 3, "Running a Verification". In this tutorial, you will verify the same package using three different methods for running a verification. You will:

   - Use the Launcher to start a verification that runs on a server.
   - Use PolySpace In One Click to start a verification that runs on a server.
   - Use the Launcher to start a verification that runs on a client.

3 Review the verification results.

   This step is in the tutorial Chapter 4, "Reviewing Verification Results".

Learning More

In this section...

Product Help

To access the help that came with your installation, select **Help > Help** or click the Help icon in the PolySpace window.

To access the online documentation for PolySpace products, go to:

`/www.mathworks.com/access/helpdesk/help/toolbox/polyspace/polyspace.html`

The MathWorks Online

For additional information and support, see:

`www.mathworks.com/products/polyspace`

Related Products

In this section...

PolySpace Products for Verifying C/C++ Code

For information about PolySpace products that verify C/C++ code, see the following:

`http://www.mathworks.com/products/polyspaceclientc/`

`http://www.mathworks.com/products/polyspaceserverc/`

PolySpace Products for Linking to Models

For information about PolySpace products that link to models, see the following:

`http://www.mathworks.com/products/polyspacemodelsl/`

`http://www.mathworks.com/products/polyspaceumlrh/`

Setting Up a Project File

About This Tutorial

In this section...
"Overview" on page 2-2
"Example Files" on page 2-2

Overview

You must have a project file before you can run a PolySpace verification of your source code. In this tutorial, you will create a project that you can use to run verifications in later tutorials.

Example Files

In this tutorial, you will verify the package `example.adb` that comes with the PolySpace installation CD. You can learn more about the files and directories required for this tutorial in "Preparing the Project Directories" on page 2-4.

Creating a New Project

In this section...

What Is a Project?

In PolySpace, a project is a named set of parameters for a verification of your software's source files. A project includes:

- The location of source files and include directories
- The location of a directory for verification results
- Analysis options

You can create your own project or use an existing one. You can create and modify a project using the Launcher graphical user interface.

A project file has one of the following file types:

Project Type	File Extension	Description
Configuration	`cfg`	Required for running a verification. Does not include generic target processors.

Project Type	File Extension	Description
PolySpace Project Model	`ppm`	Used to populate a project with analysis options, including generic target processors. Available only in PolySpace products for C/C++.
Desktop	`dsk`	Obsolete. Used in earlier versions of PolySpace software for running a verification on a client computer.

In this tutorial, you create a new project and save it as a configuration file (`.cfg`).

Preparing the Project Directories

Before you start verifying Ada code with PolySpace software, you must know the locations of the Ada source package and any other specifications upon which it may depend either directly or indirectly. You must also know where you want to store the verification results.

For each project, you decide where to store source files and results. For example, you can create a project directory and then create separate directories for the source files, include files, and results within the project directory.

For this tutorial, prepare a project directory as follows:

1. Create a project directory named `polyspace_project`.
2. Open `polyspace_project`, and create the following directories:
   - `sources`
   - `includes`
   - `results`

3 Copy the file `example.adb` from

`*Install_directory*\Examples\Demo_Ada_Single-File\sources`

to

`polyspace_project\sources`

where `*Install_directory*` is the installation directory.

4 Copy all files from

`*Install_directory*\Examples\Demo_Ada_Single-File\sources`

to

`polyspace_project\includes`.

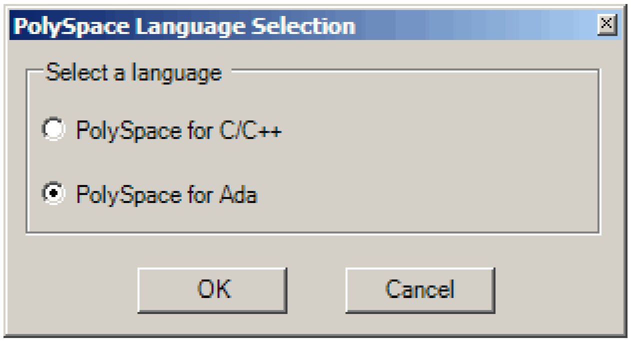
Opening the PolySpace Launcher

Use the PolySpace Launcher, a graphical user interface, to create a project and start a verification.

To open the PolySpace Launcher:

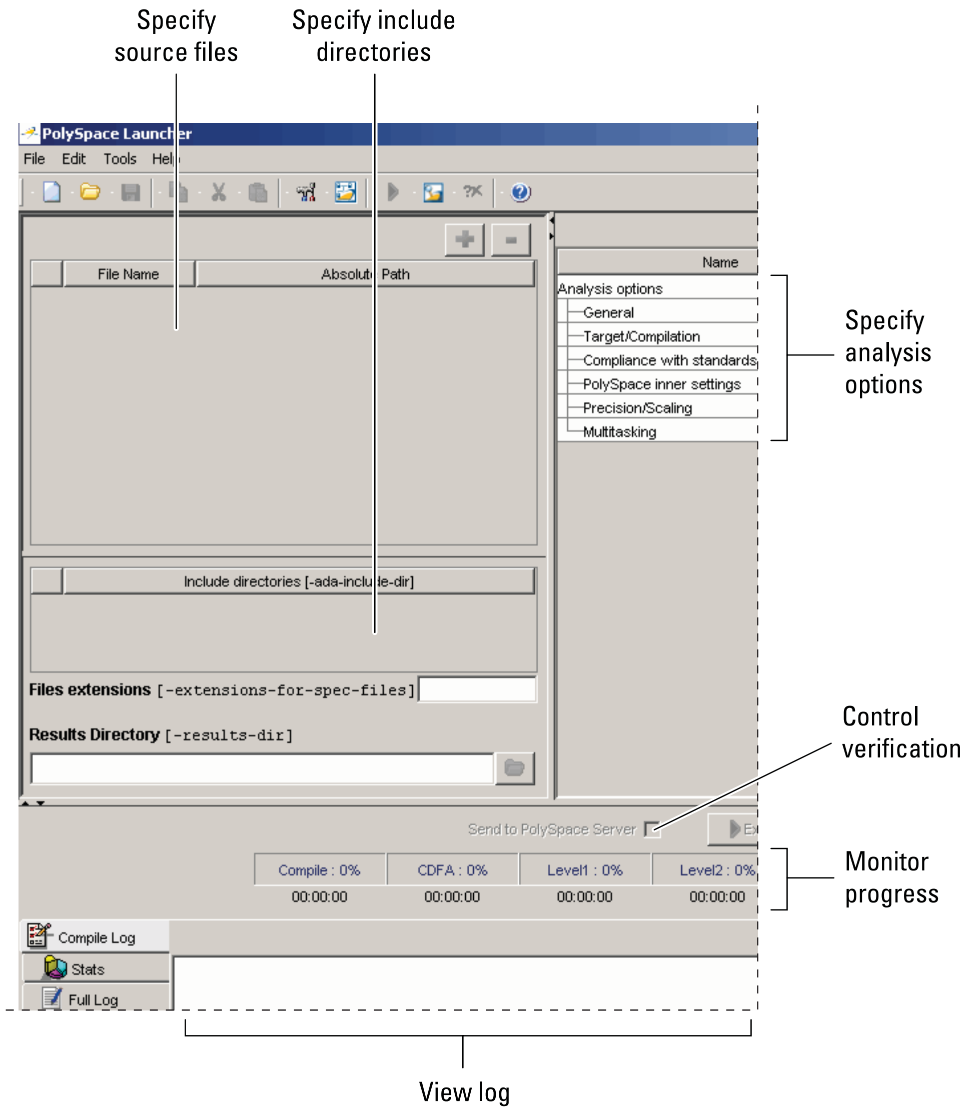
- Double-click the PolySpace Launcher icon on your desktop.

- If you have only PolySpace products for Ada installed on your computer, skip this step. If you have both PolySpace products for Ada and C/C++ on your system, the **PolySpace Language Selection** dialog box will appear.



Select **PolySpace for Ada** and click **OK**.

The PolySpace Launcher window opens.



The Launcher window has three main sections.

Use this section...	For...
Upper-left	Specifying: • Source files • Include directories • Results directory
Upper-right	Specifying analysis options
Lower	Controlling and monitoring a verification

You can resize or hide any of these sections. You learn more about the Launcher window later in this tutorial.

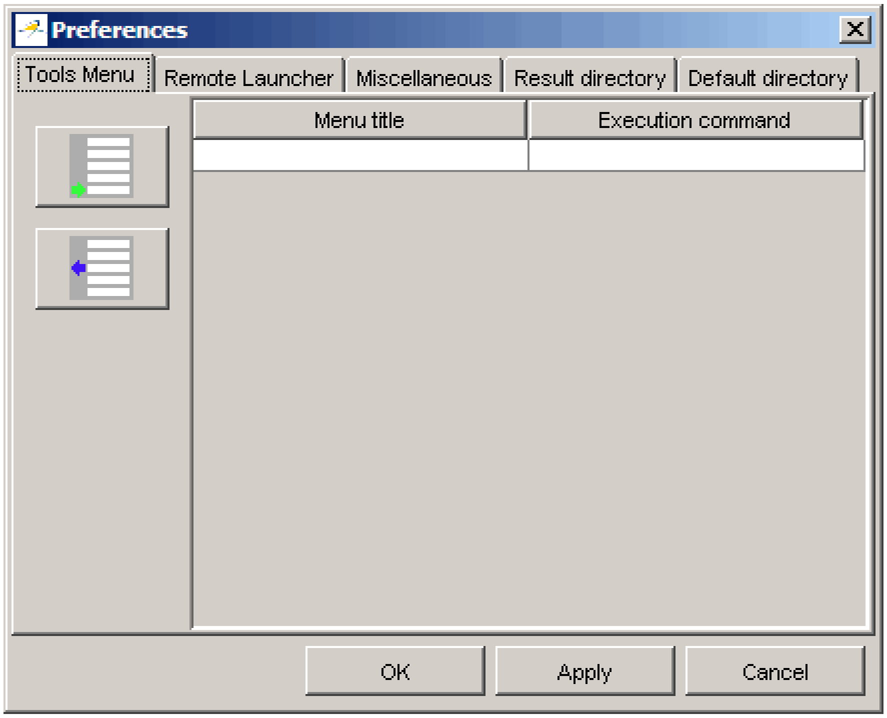
Changing the Default Directory

PolySpace software allows you to specify the default directory that appears in the directory browsers in dialog boxes. If you do not change the default directory, the default directory is the installation directory. In this tutorial, you change the default directory to the project directory that you created in "Preparing the Project Directories" on page 2-4. Changing the default directory to the project directory makes it easier for you to locate and specify source files and include directories in dialog boxes.

To change the default directory to the project directory:

1 Select **Edit > Preferences**.

   The **Preferences** dialog box appears.

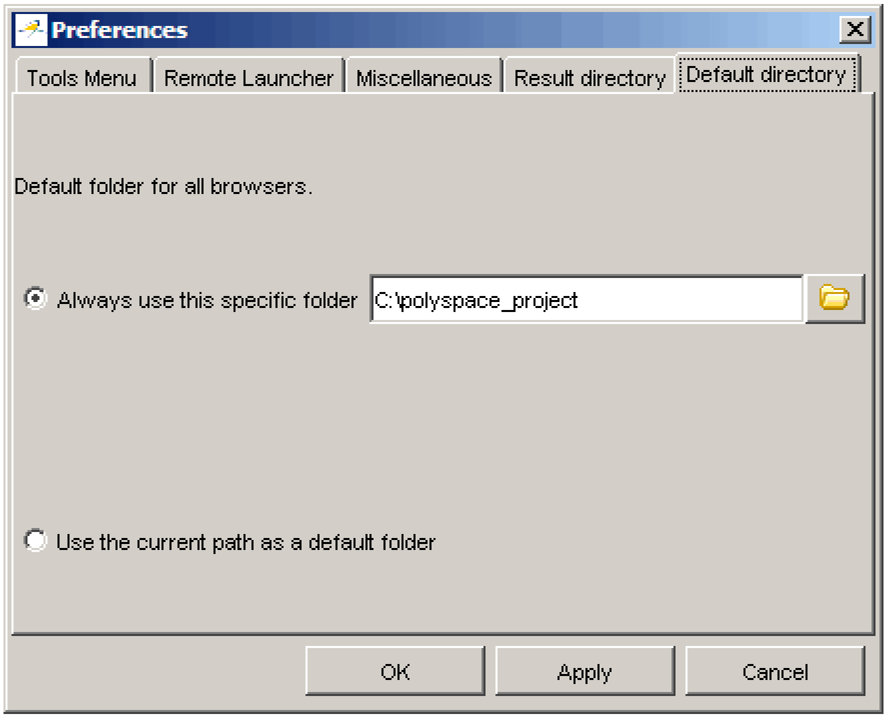


2 Select the **Default directory** tab.

3 Select **Always use this specific folder** if it is not already selected.

4 Enter or navigate to the project directory that you created earlier. In this example, the project directory is `C:\polyspace_project`.

The **Preferences** dialog box should now look like the following.



Default folder for all browsers.

[✓ Always use this specific folder] C:\polyspace_project

Use the current path as a default folder

5 Click **OK** to apply the changes and close the dialog box.

Creating a New Project to Verify an Ada Package

You must have a project, saved with file type `.cfg`, to run a verification. In this part of the tutorial, you create a new project to verify `example.adb`.

You create a new project by:

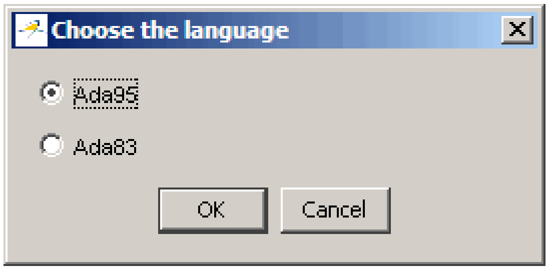
- "Opening a New project" on page 2-11
- "Specifying the Source Files, Include Directories, and Results Directory" on page 2-12
- "Specifying the Analysis Options" on page 2-15
- "Saving the Project" on page 2-16

Opening a New project

To open a new project for verifying `example.adb`:

1. Select **File > New Project**.

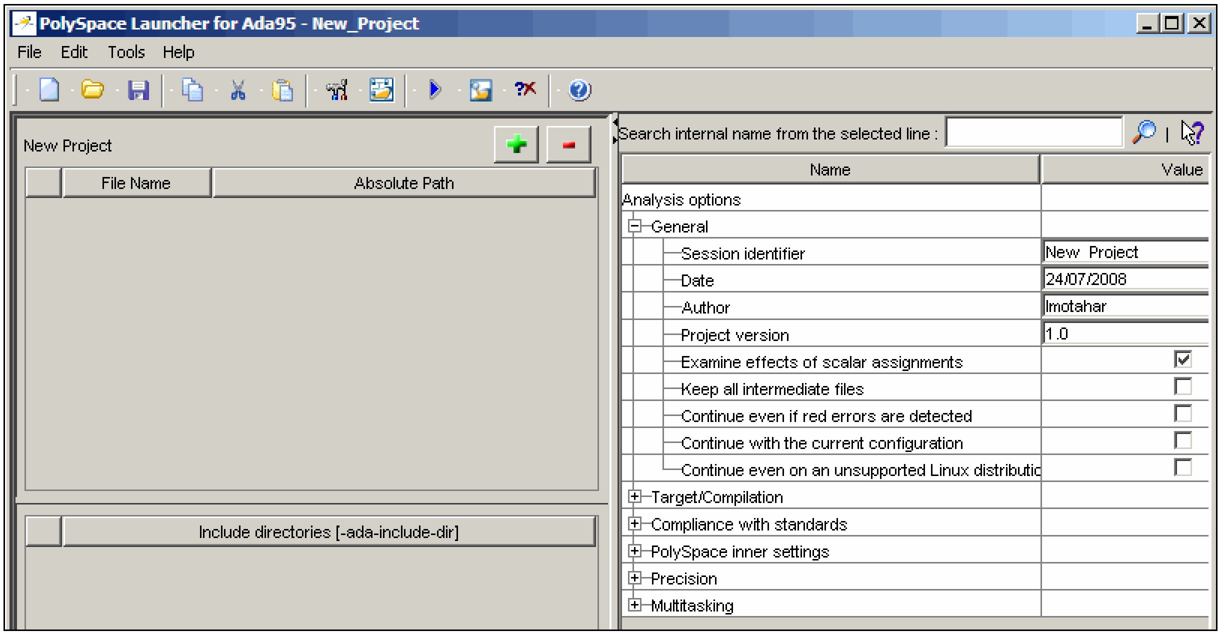
   The **Choose the language** dialog box appears:

2. Select **Ada95**, then click **OK**.

   The default project name, `New_Project`, appears in the title bar.

   In the **Analysis options** section, the **General** options node expands with default project identification information and options.


PolySpace Launcher for Ada95 - New_Project
File Edit Tools Help
New Project
File Name
Absolute Path
Include directories [-ada-include-dir]
Search internal name from the selected line :
Name
Value

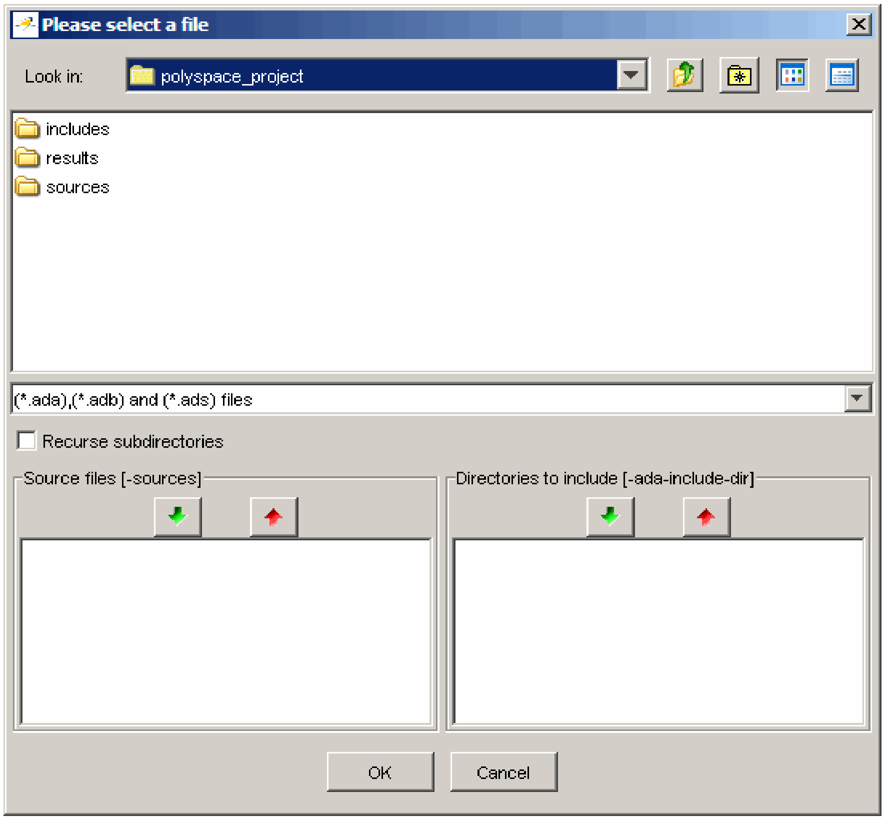
Name	Value
Analysis options	
General	
Session identifier	New Project
Date	24/07/2008
Author	lmotahar
Project version	1.0
Examine effects of scalar assignments	☑
Keep all intermediate files	☐
Continue even if red errors are detected	☐
Continue with the current configuration	☐
Continue even on an unsupported Linux distributic	☐
Target/Compilation	
Compliance with standards	
PolySpace inner settings	
Precision	
Multitasking	

Specifying the Source Files, Include Directories, and Results Directory

To specify the source files, include directories, and results directory for the verification of `example.adb`:

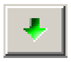
1 Click the green plus sign button in the upper right of the files section of the Launcher window.

The **Please select a file** dialog box appears.



2 The project directory `polyspace_project` should appear in the **Look in** drop-down box. If it does not, navigate to that directory.

3 Select the directory `includes` and then click the green down arrow button in the **Directories to include** section.



The path for the directory appears in the list of directories to include.

4 Double-click the directory `sources`.

5 Select the file `example.adb` and then click the green down arrow button in the **Source files** section.

The path for `example.adb` appears in the source files list.

Tip You can also drag directory and file names from an open directory directly to the source files list or the directories to include list.

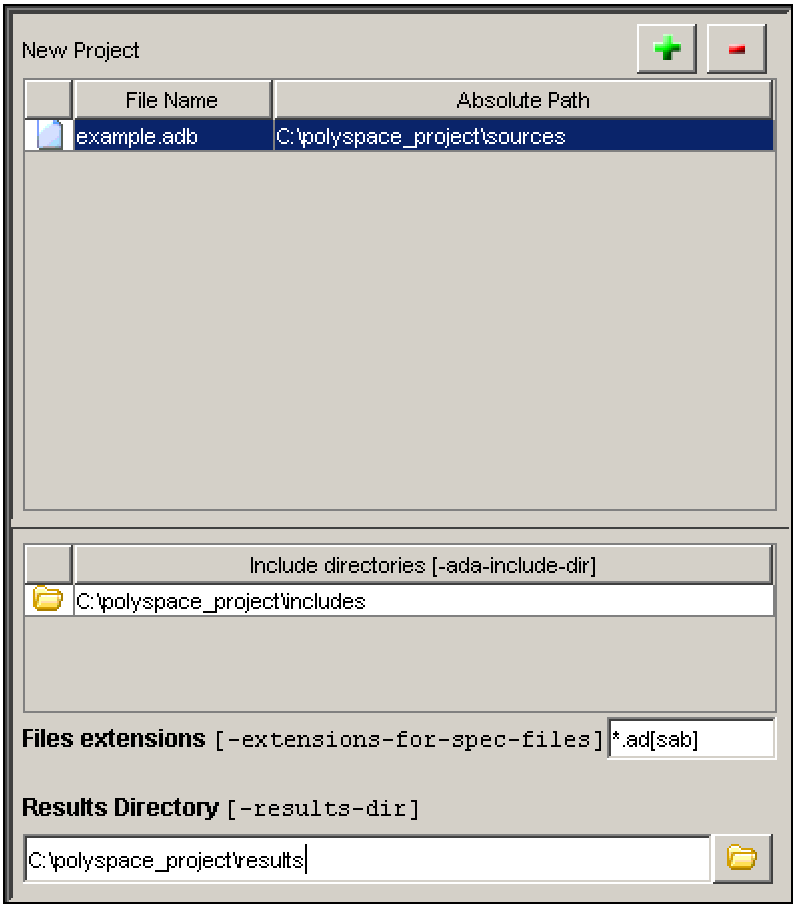
---

6 Click **OK** to apply the changes and close the dialog box.

7 In **Files extensions**, update the text in the box to: `*.ad[sab]`. This indicates that specifications for the source code may be found in `.ada`, `.ads` and `.adb` files. For example, you added the file `util.adb` to the `polyspace_project\includes` directory in the tutorial "Preparing the Project Directories" on page 2-4. If you do not update **Files extensions** to include `.adb` files, PolySpace will not recognize `util.adb` as a valid specification path. This will cause the verification process to fail.

8 In **Results Directory**, specify the directory for the verification results. Enter the path for the results directory that you created earlier. In this example, the results directory is `C:\polyspace_project\results`.

The files section in the upper left of the Launcher window now looks like this.



New Project

	File Name	Absolute Path
	example.adb	C:\polyspace_project\sources

	Include directories [-ada-include-dir]
	C:\polyspace_project\includes

Files extensions `[-extensions-for-spec-files]` *.ad[sab]

Results Directory `[-results-dir]`

C:\polyspace_project\results

Specifying the Analysis Options

The analysis options in the upper-right section of the Launcher window include identification information and parameters that PolySpace software uses during the verification process. For more information about analysis options, see “Options Description” in the *PolySpace Products for Ada Reference*.

To specify the analysis options for this tutorial:

1 In the **General** section, change the **Session identifier** to `Example_Project`.

Note The session identifier cannot contain spaces.

---

2 In the General section, select the Continue even if red errors are detected check box.

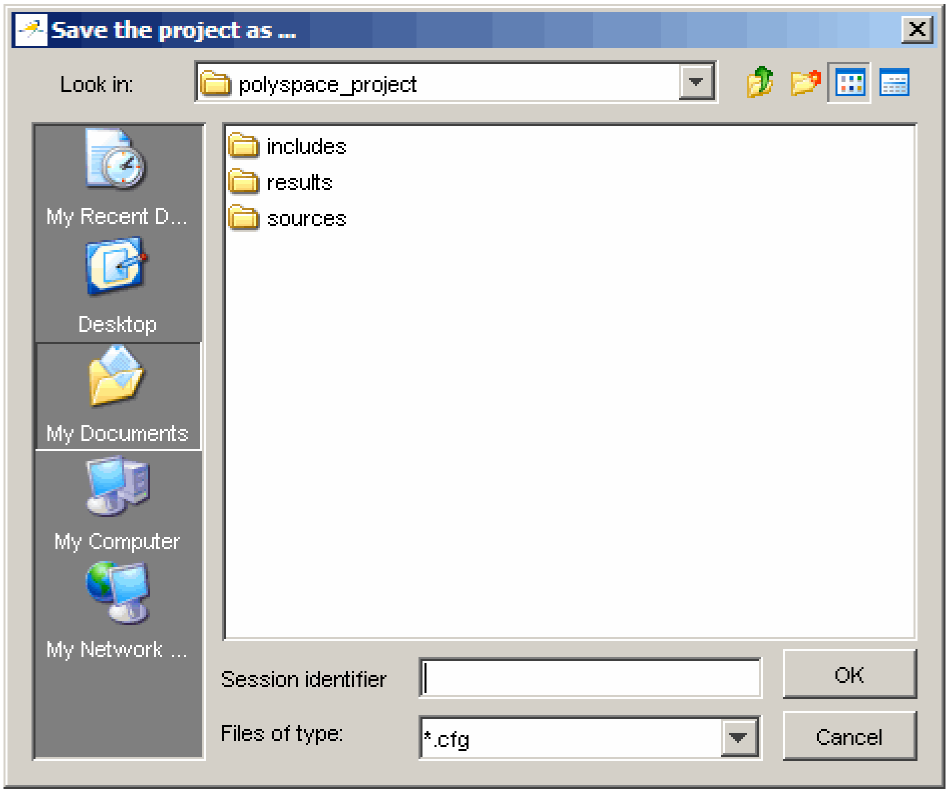
You learn about red errors in Chapter 4, “Reviewing Verification Results”.

3 Keep the default values for all other options.

Saving the Project

To save the project:

1 Select **File > Save project**. The **Save the project as** dialog box appears.



2 In **Look in**, leave the default directory, `polyspace_project`.

3 In **Session identifier**, enter `example`.

4 In **Files of type**, leave the default `*.cfg`. You must have a project file with type `cfg` to run a verification.

Note You can also run a verification with a project file of type `dsk`. Older versions of PolySpace software created files with type `dsk` for use with verifications running on a desktop PC. For more information about the `dsk` file type, see "What Is a Project?" on page 2-3.

---

5 Click **OK** to save the project and close the dialog box.

Running a Verification

About This Tutorial

In this section…

Overview

Once you have created the project `example.cfg` as described in "Creating a New Project" on page 2-3, you can run the verification.

You can run a verification on a server or a client.

Use…	For…
Server	• Best performance • Large files (more than 800 lines of code including comments) • Multitasking
Client	• An alternative to the server when the server is busy • Small files with no multitasking **Note** Verification on a client takes more time. You might not be able to use your client computer when a verification is running on it.

You can start a verification using the Launcher or using PolySpace In One Click. With either method, the verification can run on a server or a client.

Use...	For...
Launcher	A basic way to start a verification. You specify the source files in the project file. With the project file open, you click a button to start the verification.
PolySpace In One Click	A convenient way to start the verification of several files which use the same verification options. Once you specify the project file containing the verification options, you specify the source files by selecting them from a Microsoft Windows folder. You start the verification by sending the selected files to PolySpace software.

In this tutorial, you learn how to run a verification on a server and on a client, and you learn how to start a verification using the Launcher and using PolySpace In One Click. You verify the package `example.adb` three times using a different method each time. You use:

1. The Launcher to start a verification that runs on a server.
2. PolySpace In One Click to start a verification that runs on a server.
3. The Launcher to start a verification that runs on a client.

Each verification stores the same results in `polyspace_project\results`. You review these results in the tutorial Chapter 4, “Reviewing Verification Results”.

Before You Start

Before you start this tutorial, you must complete Chapter 2, “Setting Up a Project File”. You use the directories and project file, `example.cfg`, from that tutorial to run the verifications.

Opening the Project

To run a verification, you must have an open project file. For this tutorial, you use the project file `example.cfg` that you created in Chapter 2, "Setting Up a Project File". Open `example.cfg` if it is not already open.

To open `example.cfg`:

1. If the PolySpace Launcher is not already open, open it by double-clicking the PolySpace Launcher icon.
2. Select **File > Open project**.

   The **Please select a file** dialog box opens.
3. In **Look in**, navigate to `polyspace_project`.
4. Select `example.cfg`.
5. Click **Open** to open the file and close the dialog box.

Using the Launcher to Start a Verification That Runs on a Server

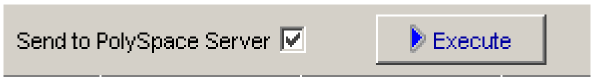
In this section...

Starting the Verification

In this part of the tutorial, you run the verification on a server.

To start a verification that runs on a server:

1 Select the **Send to PolySpace Server** check box next to the **Execute** button in the middle of the Launcher window.



Note If you select **Set this option to use the server mode by default in every new project** in the Remote Launcher pane of the preferences, the **Send to PolySpace Server** check box is selected by default when you create a new project.

2 Click **Execute**.

Note If you see the message `Verification process failed`, click **OK** and go to "Troubleshooting a Failed Verification" on page 3-12.

The verification has three main phases:

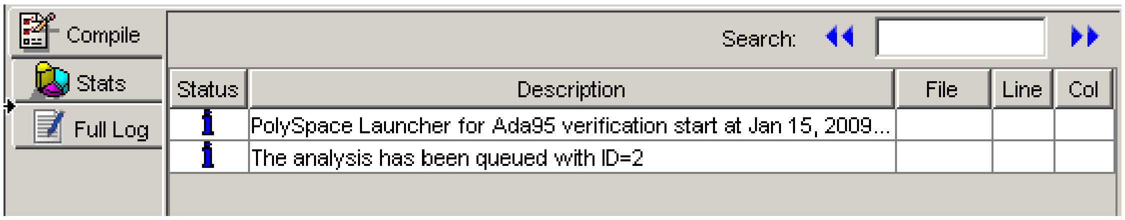
a. Checking syntax and semantics (the compile phase). Because PolySpace software is independent of any particular Ada compiler, it ensures that your code is portable, maintainable, and complies with ANSI® standards.

b. Generating a main if it does not find a main and the **Generate a Main** option is selected. For more information about generating a main, see the section "-main-generator" in the "Options Description" chapter of the *PolySpace Client/Server for Ada User's Guide*.

c. Analyzing the code for run-time errors and generating color-coded diagnostics.

The compile phase of the verification runs on the client. When the compile phase finishes:

- A message dialog box tells you that the verification is completed. This message means that the part of the verification that takes place on the client is complete. The rest of the verification runs on the server.
- A message in the log area tells you that the verification was transferred to the server and gives you the identification number (Analysis ID) for the verification. For this verification, the identification number is `2`.



Status	Description	File	Line	Col
i	PolySpace Launcher for Ada95 verification start at Jan 15, 2009...			
i	The analysis has been queued with ID=2			

3. When you see the message `Verification process completed`, click **OK** to close the message dialog box.

4. Stop the Launcher by clicking **File > Quit**.

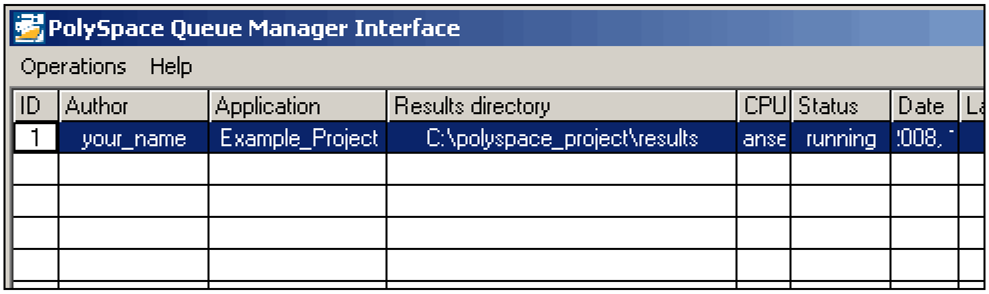
Monitoring the Progress of the Verification

You monitor the progress of the verification using the PolySpace Queue Manager (also called the Spooler).

To monitor the verification of `Example_Project`:

1 Double-click the **PolySpace Spooler** icon:

The **PolySpace Queue Manager Interface** opens.



PolySpace Queue Manager Interface

Operations Help

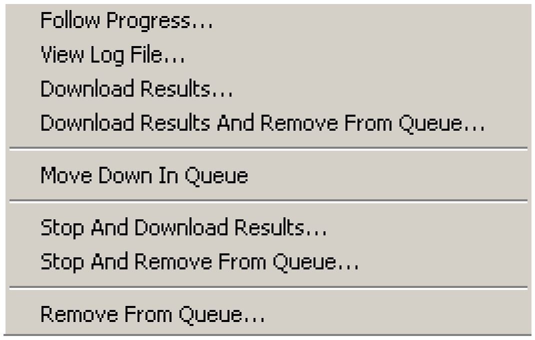
ID	Author	Application	Results directory	CPU	Status	Date	La
1	your_name	Example_Project	C:\polyspace_project\results	anse	running	'008, '	

Tip You can also open the Polyspace Queue Manager Interface by clicking the PolySpace Queue Manager icon in the PolySpace Launcher toolbar.

---

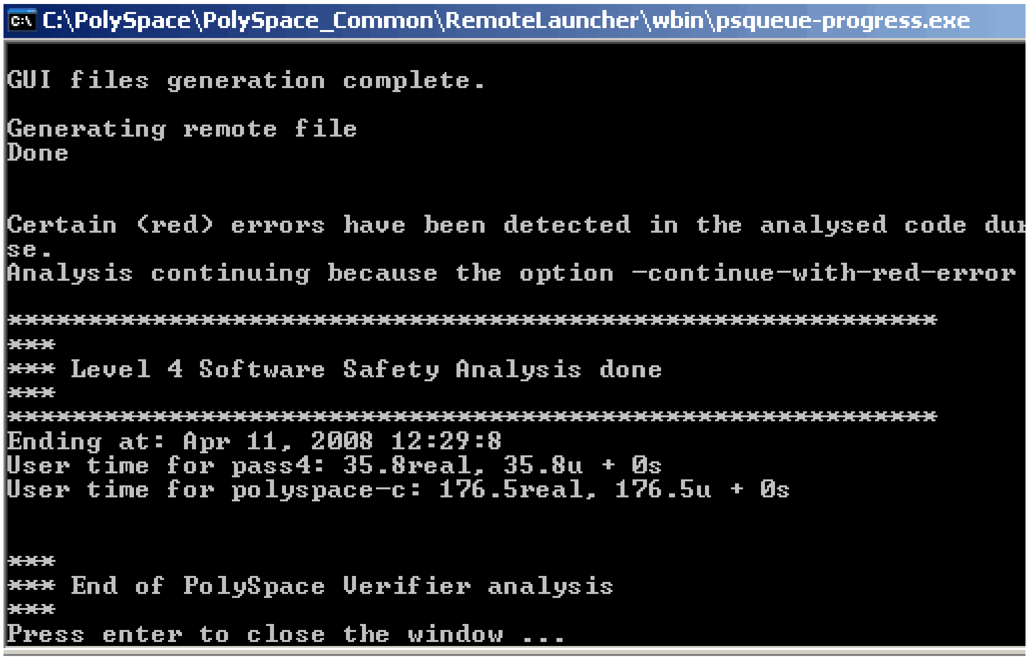
2 Point anywhere in the row for ID 1.

3 Right-click to open the context menu for this verification.


Follow Progress...
View Log File...
Download Results...
Download Results And Remove From Queue...
Move Down In Queue
Stop And Download Results...
Stop And Remove From Queue...
Remove From Queue...

4 Select **View log file**.

A window opens displaying the last one-hundred lines of the verification.

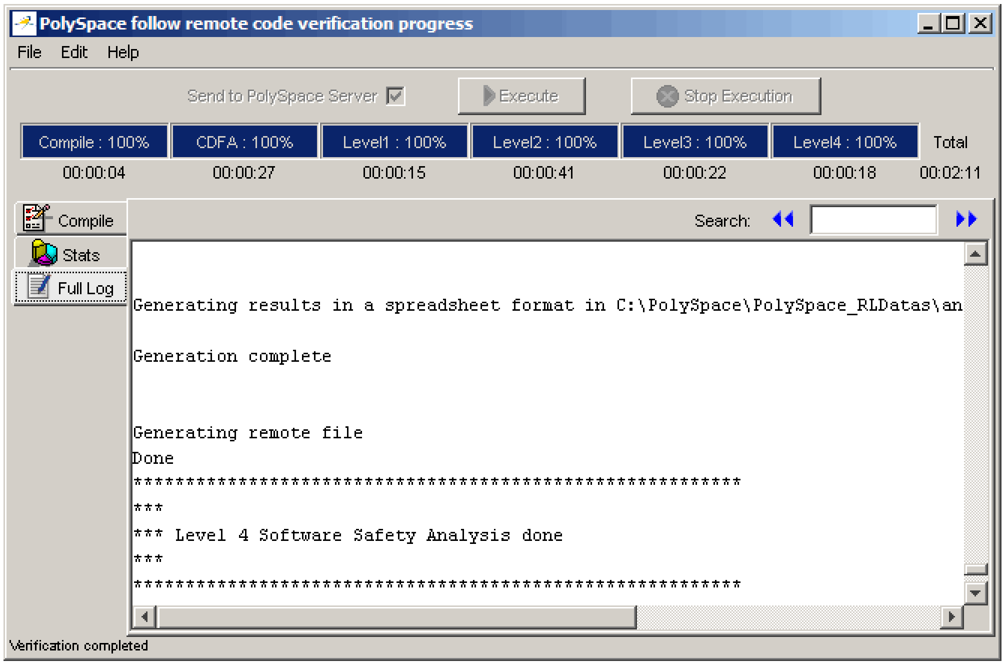


```
C:\PolySpace\PolySpace_Common\RemoteLauncher\wbin\psqueue-progress.exe

GUI files generation complete.

Generating remote file
Done


Certain (red) errors have been detected in the analysed code dur
se.
Analysis continuing because the option -continue-with-red-error

**********************************************************
***
*** Level 4 Software Safety Analysis done
***
**********************************************************
Ending at: Apr 11, 2008 12:29:8
User time for pass4: 35.8real, 35.8u + 0s
User time for polyspace-c: 176.5real, 176.5u + 0s


***
*** End of PolySpace Verifier analysis
***
Press enter to close the window ...
```

5 Press **Enter** to close the window.

6 Select **Follow Progress** from the context menu.

A Launcher window labeled **PolySpace follow remote analysis progress for Ada** appears.



```
Generating results in a spreadsheet format in C:\PolySpace\PolySpace_RLDatas\an

Generation complete


Generating remote file
Done
**********************************************************
***
*** Level 4 Software Safety Analysis done
***
**********************************************************
```

Verification completed

You can monitor the progress of the verification by watching the progress bar and viewing the logs at the bottom of the window. The word `processing` appears under the current phase. The progress bar highlights each completed phase and displays the amount of time for that phase.

The logs report additional information about the progress of the verification. The information appears in the log display area at the bottom of the window. The full log displays by default. It display messages, errors, and statistics for all phases of the verification. You can search the full log by entering a search term in the **Search in the log** box and clicking the left arrows to search backward or the right arrows to search forward.

7 Click the **Compile Log** button to display compile phase messages and errors. You can search the log by entering search terms in the **Search in the log** box and clicking the left arrows to search backward or the right arrows to search forward.

8 Click the **Stats** button to display statistics, such as analysis options, stubbed functions, and the verification checks performed.

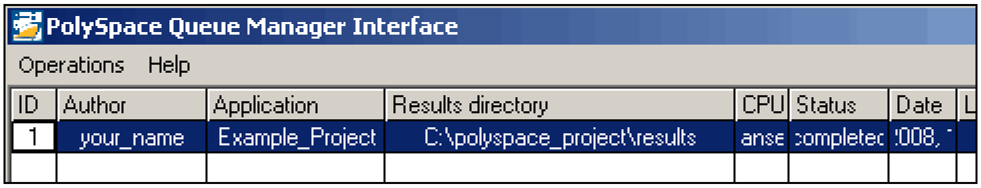
9 Click the refresh button

to update the stats log display as the verification progresses.

10 Select **File > Quit** to close the progress window.

11 Wait for the verification to finish.

When the verification is complete, the status in the **PolySpace Queue Manager Interface** changes from `running` to `completed`.



PolySpace Queue Manager Interface

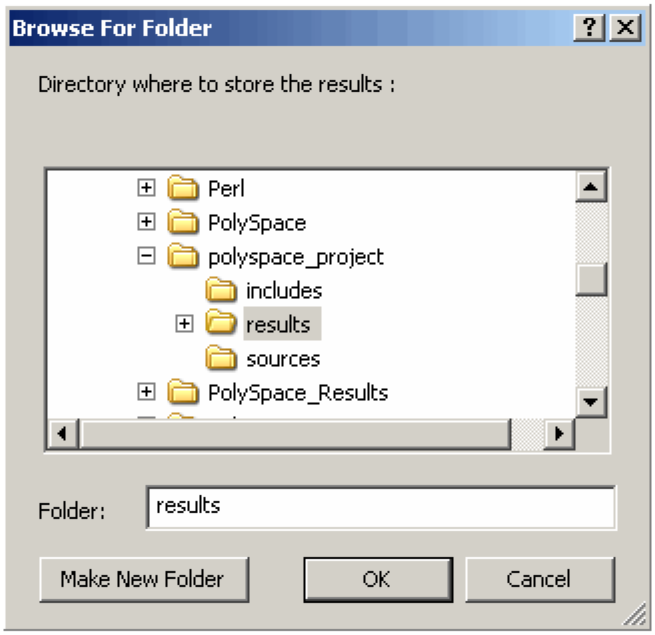
ID	Author	Application	Results directory	CPU	Status	Date	L
1	your_name	Example_Project	C:\polyspace_project\results	anse	:omplete(	'008, '	

Downloading Results from the Server to the Client

At the end of the verification, the results are on the server. To download the results to your client:

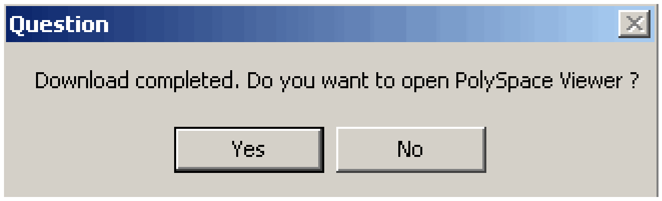
1 In the **PolySpace Queue Manager Interface**, select **Download Results** from the context menu for the verification.

The **Browse For Folder** dialog box appears with the `polyspace_project\results` folder selected.



2 Click **OK** to close the dialog box.

A dialog box appears telling you that the download is complete and asking if you want to open the PolySpace Viewer.

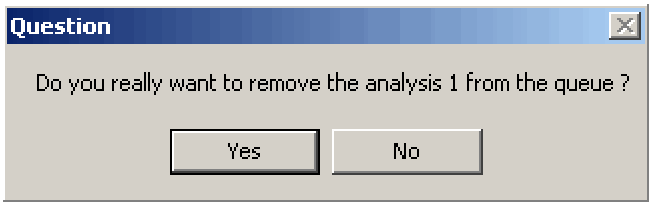


Download completed. Do you want to open PolySpace Viewer ?

3 Click **No**.

4 Select **Remove From Queue** from the context menu.

A dialog box appears asking you to confirm that you want to remove the verification from the queue.



Do you really want to remove the analysis 1 from the queue ?

5 Click **Yes**.

Note

- To download the results and remove the verification from the queue, select **Download Results And Remove From Queue** from the context menu.
- If you download results before the verification completes, you get partial results and the verification continues.

---

6 Select **Operations > Exit** to close the **PolySpace Queue Manager Interface**.

Once the results are on your client, you can review them using the PolySpace Viewer. You review the results from the verification in Chapter 4, "Reviewing Verification Results".

Troubleshooting a Failed Verification

When you see a message that the verification failed, it indicates that PolySpace software could not perform the verification. The following sections present some possible reasons for a failed verification.

Hardware Does Not Meet Requirements

The verification fails if your computer does not have the minimal hardware requirements. For information about the hardware requirements, see

`www.mathworks.com/products/polyspaceclientada/requirements.html`.

To determine if this is the cause of the failed verification, search the log for the message:

```
Errors found when verifying host configuration.
```

You can:

- Upgrade your computer to meet the minimal requirements.
- Select the **Continue with current configuration option** in the General section of the Analysis options and run the verification again.

You Did Not Specify the Location of Included Files

If you see a message in the log, such as the following, either the files are missing or you did not specify the location of included files.

```
Verifier found an error in example.adb:23:14:  "runtime_error
(spec)" depends on "types (spec)"
```

For information on how to specify the location of include files, see "Creating a New Project to Verify an Ada Package" on page 2-10.

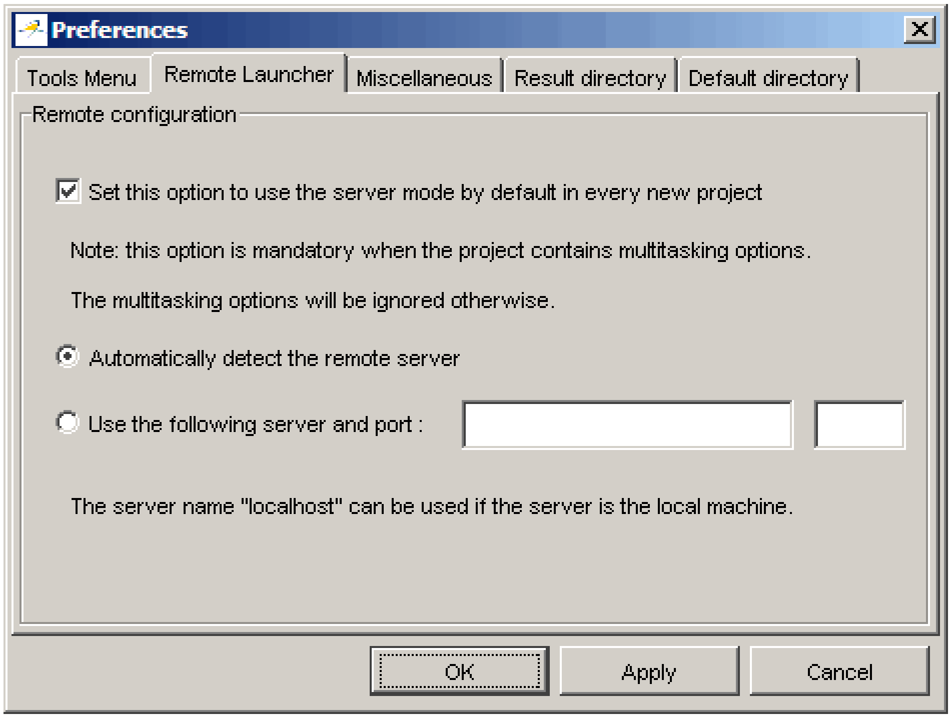
PolySpace Software Cannot Find the Server

If you see the following message in the log, PolySpace software cannot find the server.

```
Error:  Unknown host :
```

PolySpace software uses information in the preferences to locate the server. To find the server information in the preferences:

1. Select **Edit > Preferences**.
2. Select the **Remote Launcher** tab.



Remote configuration

[✓] Set this option to use the server mode by default in every new project

Note: this option is mandatory when the project contains multitasking options.

The multitasking options will be ignored otherwise.

[● Automatically detect the remote server]

○ Use the following server and port :

The server name "localhost" can be used if the server is the local machine.

By default, PolySpace software automatically finds the server. You can specify the server by selecting **Use the following server and port** and providing the server name and port. For information about setting up a server, see the *PolySpace Installation Guide*.

Using PolySpace In One Click to Start a Verification That Runs on a Server

In this section...

Overview of PolySpace In One Click

In a Microsoft Windows environment, PolySpace software provides a convenient way to streamline your work when you want to verify several packages using the same set of options. Once you have set up a project file that has the options you want, you designate that project as the *active project*, and then send the source files to PolySpace software for verification. You do not have to update the project with source file information. This process is called *PolySpace In One Click*.

In this part of the tutorial, using PolySpace In One Click, you learn how to:

1. Set the active project.

2. Send source files to PolySpace software for verification.

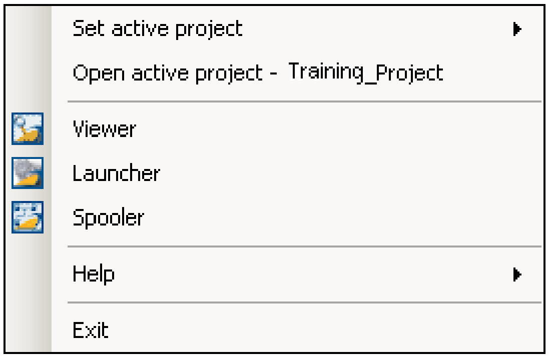
Setting the Active Project

The active project is the project that PolySpace In One Click uses to verify the packages that you select. Once you have set an active project, it remains active until you change the active project. PolySpace software uses the analysis options from the project; it does not use the source files or results directory from the project.

To set the active project:

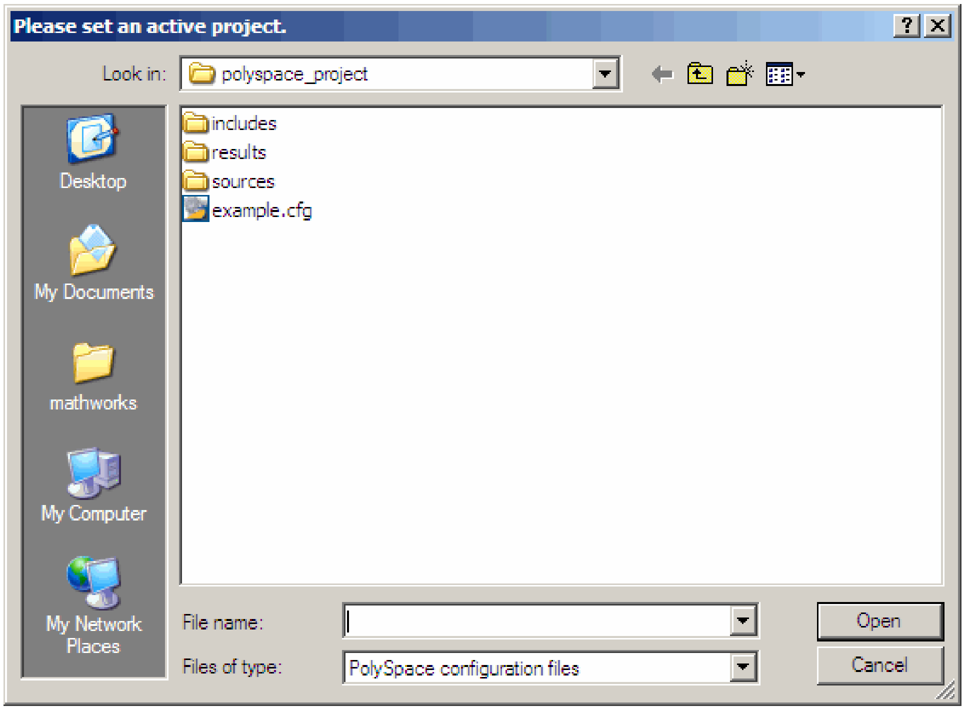
1. Right-click the PolySpace In One Click icon in the taskbar area of your Windows desktop:

The context menu appears.



2 Select **Set active project > Browse** from the menu.

The **Please set an active project** dialog box appears:



3 In **Look in**, navigate to `polyspace_project`.

4 Select `example.cfg`.

5 Click **Open** to apply the changes and close the dialog box.

Sending the Files to PolySpace Software

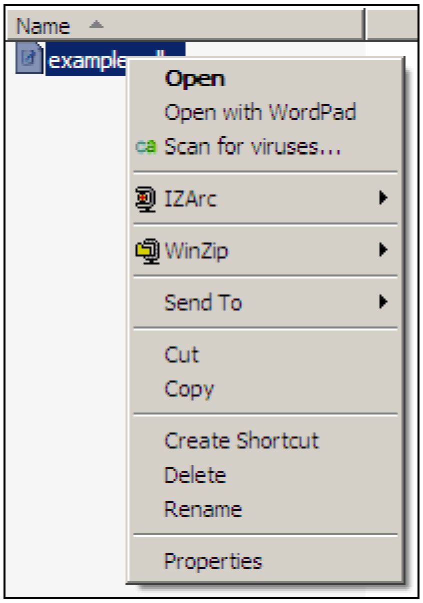
You can send several packages to PolySpace software for verification. For this tutorial, you send one package, `example.adb`.

To send `example.adb` to PolySpace software for verification:

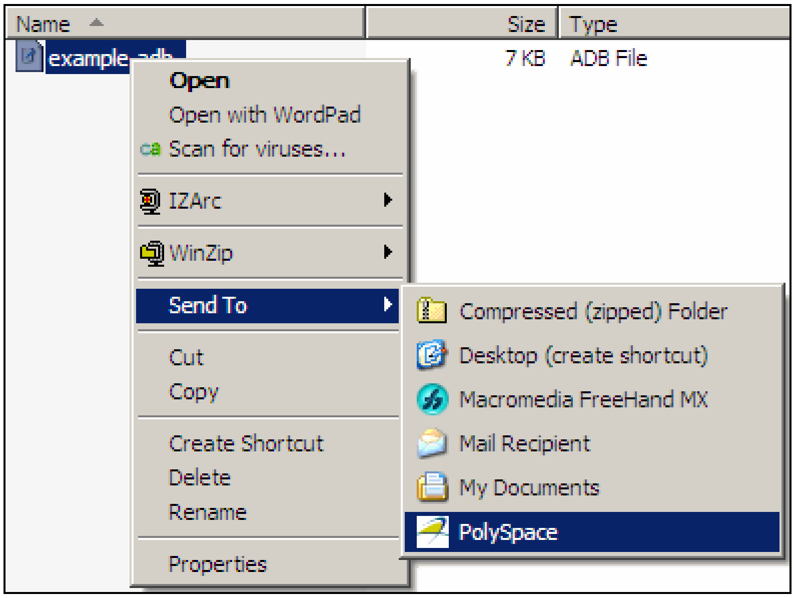
1 Navigate to the directory `polyspace_project\sources`.

2 Right-click the file `example.adb`.

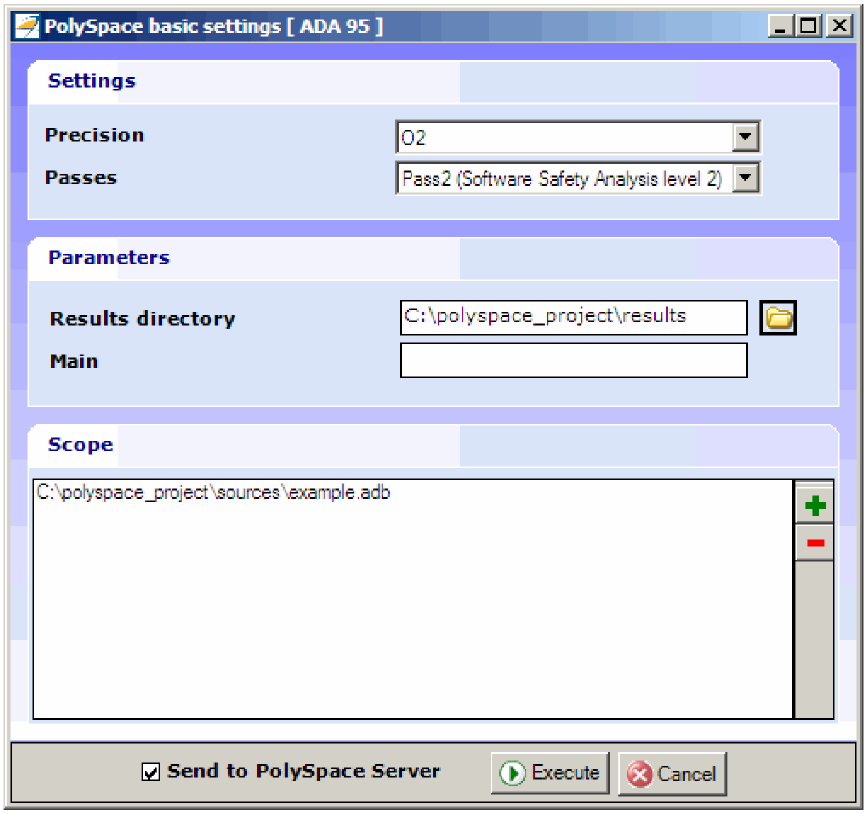
The context menu appears.



3 Select **Send To > PolySpace**.



The **PolySpace basic settings** dialog box appears.



Precision: O2
Passes: Pass2 (Software Safety Analysis level 2)
Results directory: C:\polyspace_project\results
Main:
C:\polyspace_project\sources\example.adb
☑ Send to PolySpace Server

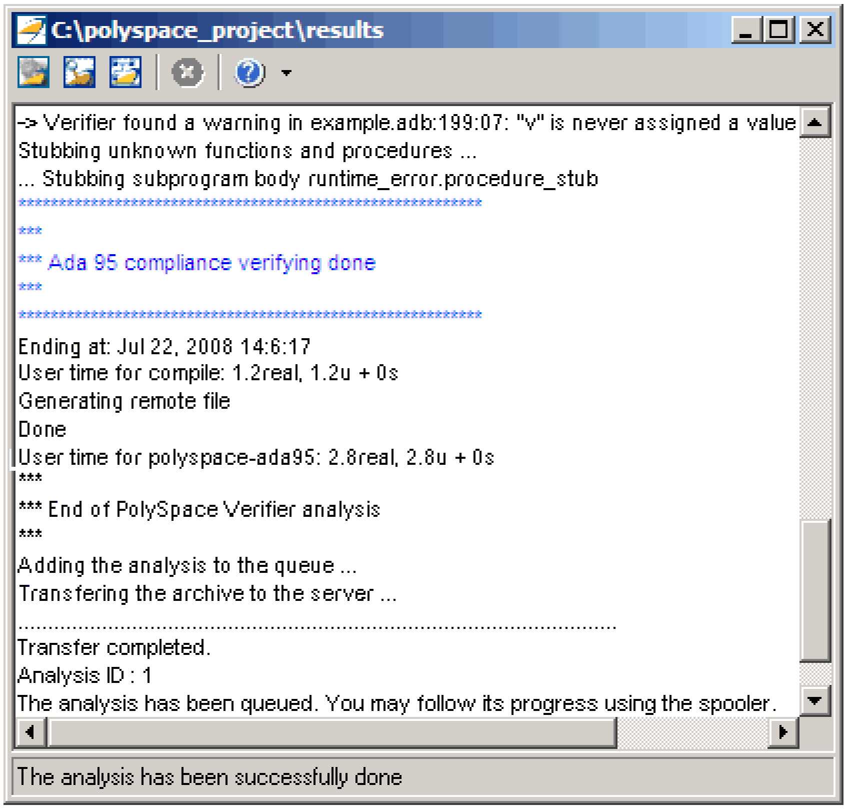
4 Make sure that **Results directory** is `polyspace_project\results`.

5 Select the **Send to PolySpace Server** option if it is not already selected.

6 Leave the default values for the other parameters.

Click **Execute**.

The verification log appears.



```
-> Verifier found a warning in example.adb:199:07: "v" is never assigned a value
Stubbing unknown functions and procedures ...
... Stubbing subprogram body runtime_error.procedure_stub
***********************************************************
***
*** Ada 95 compliance verifying done
***
***********************************************************
Ending at: Jul 22, 2008 14:6:17
User time for compile: 1.2real, 1.2u + 0s
Generating remote file
Done
User time for polyspace-ada95: 2.8real, 2.8u + 0s
***
*** End of PolySpace Verifier analysis
***
Adding the analysis to the queue ...
Transfering the archive to the server ...
.............................................................................................
Transfer completed.
Analysis ID : 1
The analysis has been queued. You may follow its progress using the spooler.
```

The analysis has been successfully done

The compile phase of the verification runs on the client. When the compile phase completes:

- You see the message:

  `End of PolySpace Verifier analysis`

- A message in the log area tells you that the verification was transferred to the server and gives you the identification number (Analysis ID) for the verification. For this verification, the identification number is `1`.
- Monitor the verification using the Spooler. For information on using the Spooler to monitor a verification on a server, see "Monitoring the Progress of the Verification" on page 3-6.

- When the verification completes, download the results to `polyspace_project\results`. For information on downloading results from a server to a client, see “Downloading Results from the Server to the Client” on page 3-10

  You review the results in Chapter 4, “Reviewing Verification Results”.

Using the Launcher to Start a Verification That Runs on a Client

In this section...

Starting the Verification

For the best performance, run verifications on a server. If the server is busy or you want to verify a small package, you can run a verification on a client.

Note Because a verification on a client can process only a limited number of variable assignments and function calls, the source code should have no more than 800 lines of code.

To start a verification that runs on a client:

1. Open the Launcher if it is not already open.

2. Open the project file `example.cfg` if it is not already open.

   For information about opening a project file, see "Opening the Project" on page 3-4.

3. Make sure that the **Send to PolySpace Server** check box is clear.

4. If you see a warning that multitasking is not available when you run a verification on the client, click **OK** to continue and close the message box. This warning only appears when you clear the **Send to PolySpace Server** check box.

5. Click the **Execute** button.

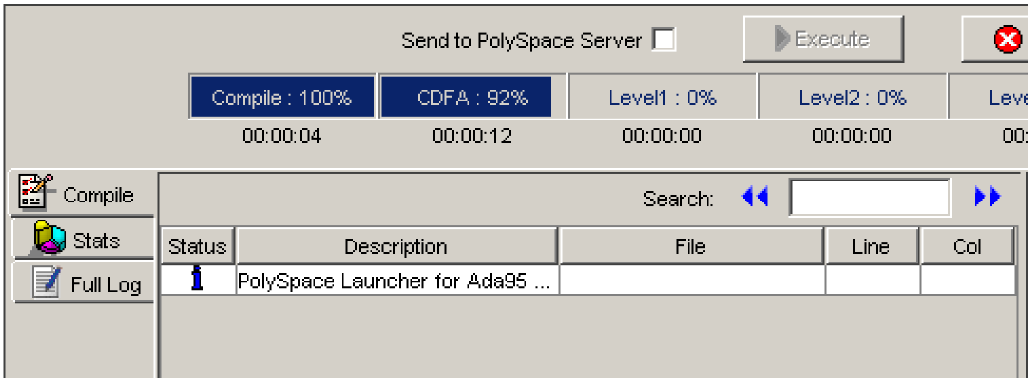
6 If you see a caution that PolySpace software will remove existing results from the results directory, click **Yes** to continue and close the message dialog box.

The progress bar and logs area of the Launcher window become active.

Note If you see the message `Verification process failed`, click **OK** and go to "Troubleshooting a Failed Verification" on page 3-12.

Monitoring the Progress of the Verification

You can monitor the progress of the verification by watching the progress bar and viewing the logs at the bottom of the Launcher window.



Status	Description	File	Line	Col
i	PolySpace Launcher for Ada95 ...			

The progress bar highlights the current phase in blue and displays the amount of time and completion percentage for that phase.

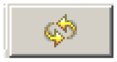
The logs report additional information about the progress of the verification. To view a log, click the button for that log. The information appears in the log display area at the bottom of the Launcher window. Follow the next steps to view the logs:

1 The compile log displays by default.

This log displays compile phase messages and errors. You can search the log by entering search terms in the **Search in the log** box and clicking the left arrows to search backward or the right arrows to search forward.

2. Click the **Stats** button to display statistics, such as analysis options, stubbed functions, and the verification checks performed.

3. Click the refresh button

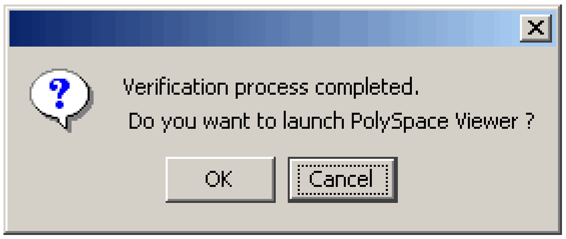
   to update the display as the verification progresses.

4. Click the **Full Log** button to display messages, errors, and statistics for all phases of the verification.

   You can search the full log by entering a search term in the **Search in the log** box and clicking the left arrows to search backward or the right arrows to search forward.

Completing the Verification and Stopping the Launcher

When the verification completes, a message dialog box appears telling you that the verification is complete and asking if you want to open the Viewer. For this tutorial, do not open the Viewer at this point.



Verification process completed.
Do you want to launch PolySpace Viewer ?

To indicate that you do not want to open the Viewer:

- Click **Cancel**.

You can also open the Viewer from the Launcher toolbar, but for this tutorial, you do not do this. For this tutorial, close the Launcher.

To close the Launcher:

- Select **File > Quit**.

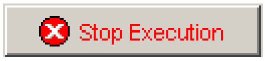
In the tutorial Chapter 4, "Reviewing Verification Results", you open the Viewer and review the verification results.

Stopping the Verification Before It Completes

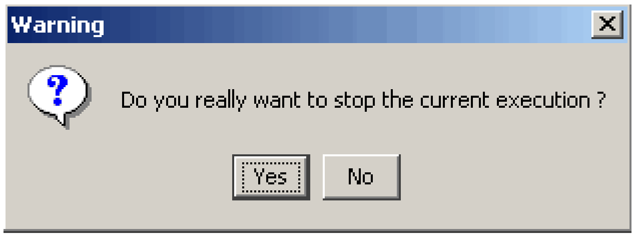
You can stop the verification before it completes. If you stop the verification, results will be incomplete, and if you start another verification, the verification starts over from the beginning.

To stop a verification:

1. Click the **Stop Execution** button.



A warning dialog box appears.



Do you really want to stop the current execution ?

2. Click **Yes**.

The verification stops and the message `Verification process stopped` appears.

3. Click **OK** to close the **Message** dialog box.

Note Closing the Launcher window does *not* stop the verification. To resume display of the verification progress, open the Launcher window and open the project that you were verifying when you closed the Launcher window.

Reviewing Verification Results

About This Tutorial

In this section...
"Overview" on page 4-2
"Before You Start" on page 4-2

Overview

In the previous tutorial, Chapter 3, "Running a Verification", you completed a verification of the package `example.adb`. In this tutorial, you explore the verification results.

PolySpace software provides a graphical user interface, called the Viewer, that you use to review results. In this tutorial, you learn:

1. How to use the Viewer, including how to:
   - Open the Viewer and open verification results.
   - Select the Viewer mode.
   - Explore results in expert mode.
   - Explore results in assistant mode.
   - Generate reports.
2. How to interpret the color-coding that PolySpace software uses to identify the severity of an error.
3. How to find the location of an error in the source code.

Before You Start

Before starting this tutorial, complete the tutorial Chapter 3, "Running a Verification". In this tutorial, you use the verification results stored in this file:

`polyspace_project\results\RTE_px_O2_Example_Project_LAST_RESULTS.rte`.

Opening the Viewer and the Verification Results

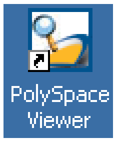
In this section...

Opening the Viewer

You use the Viewer to review verification results. Open the Viewer if it is not already open.

To open the Viewer:

- Double-click the PolySpace Viewer icon:



Note You can also open the Viewer from the Launcher by clicking the Viewer icon in the Launcher toolbar with or without an open project.

Selecting the Viewer Mode

You can review verification results in *expert* mode or *assistant* mode:

- In expert mode, you decide how you review the results.
- In assistant mode, PolySpace software guides you through the results.

You switch from one mode to the other by clicking a button in the Viewer toolbar. For this part of the tutorial, the Viewer should be in expert mode. If the Viewer is in expert mode, the switch mode button in the toolbar displays **Assistant**.

If the Viewer is not in expert mode, click the mode button to switch to expert mode.

You learn more about expert and assistant modes later in this tutorial.

Opening the Results

To open the verification results:

1. Select **File** > **Open**.
2. In the **Please select a file dialog box**, navigate to `polyspace_project\results` and select the file `RTE_px_02_Example_Project_LAST_RESULTS.rte`.
3. Click the **Open** button.

   The results appear in the Viewer window.

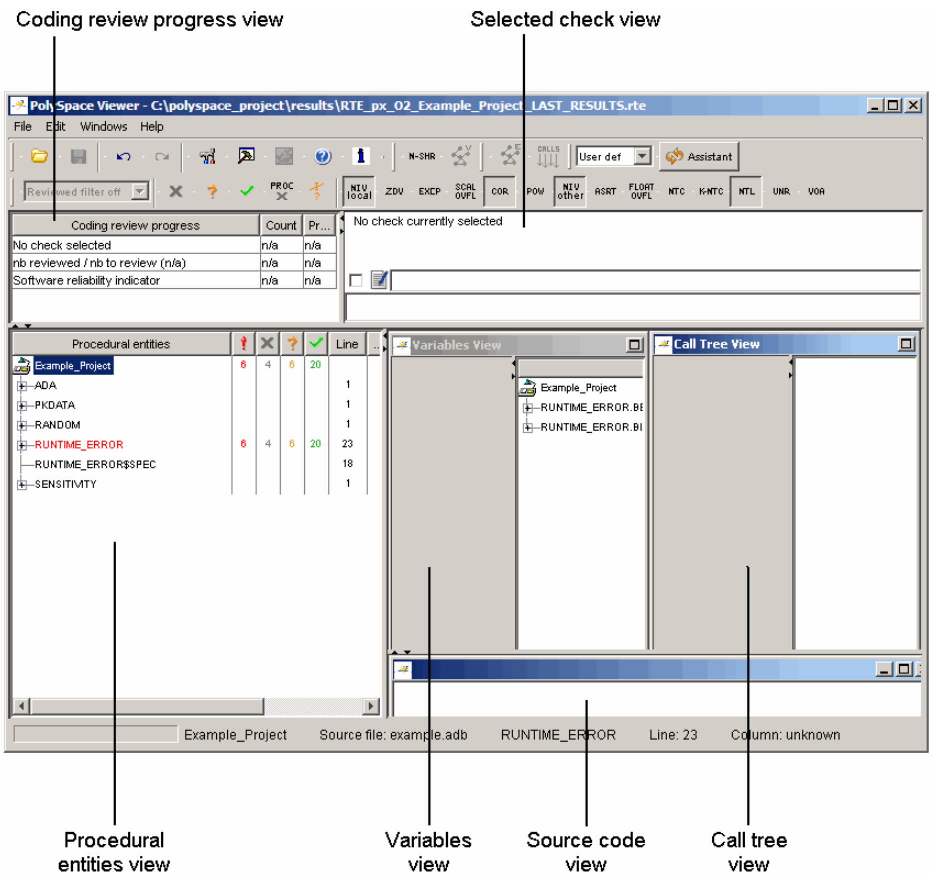
Note The file `RTE_px_02_Example_Project_LAST_RESULTS.rte` represents the verification with the highest level of precision. The lower level results files that you see in the `polyspace_project\results` directory represent lower precision verifications.

Exploring the Viewer Window

In this section...
"Overview" on page 4-5
"Reviewing the Procedural Entities View" on page 4-7

Overview

The PolySpace Viewer window looks like this.


Coding review progress view
Selected check view
PolySpace Viewer - C:\polyspace_project\results\RTE_px_02_Example_Project_LAST_RESULTS.rte
File Edit Windows Help
N-SHR
User def
Assistant
Reviewed filter off
PROC
NIV local ZDV EXCP SCAL OVFL COR POW NIV other ASRT FLOAT OVFL NTC K-NTC NTL UNR VOA
Variables View
Call Tree View
Procedural entities view
Variables view
Source code view
Call tree view

Coding review progress	Count	Pr...
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

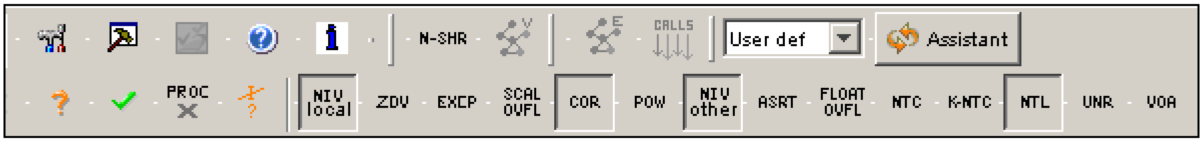
No check currently selected

Procedural entities	!	✗	?	✓	Line
Example_Project	6	4	6	20	
ADA					1
PKDATA					1
RANDOM					1
RUNTIME_ERROR	6	4	6	20	23
RUNTIME_ERROR$SPEC					18
SENSITIVITY					1

Example_Project
RUNTIME_ERROR.BE
RUNTIME_ERROR.BI

Example_Project Source file: example.adb RUNTIME_ERROR Line: 23 Column: unknown

The appearance of the Viewer toolbar depends on the Viewer mode. Because the Viewer is in expert mode, the expert mode toolbar is displayed.



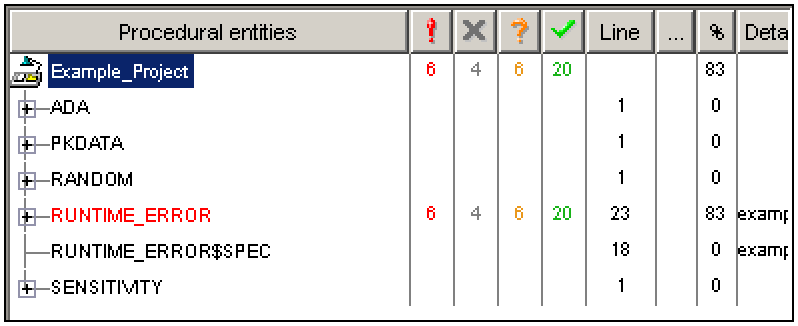
In both expert mode and assistant mode, the Viewer window has six sections below the toolbar. Each section provides a different view of the results. The following table describes these views.

This view...	Displays...
Procedural entities view (lower left)	List of the diagnostics (checks) for each file and function in the project
Source code view (lower right)	Source code for a selected check in the procedural entities view
Coding review progress view (upper left)	Statistics about the review progress for checks with the same type and category as the selected check
Selected check view (upper right)	Details about the selected check
Variables view	Information about the global variables declared in the source code **Note** The file that you use in this tutorial does not have global variables.
Call tree view	Tree structure of function calls

You can resize or hide any of these sections. You learn more about the Viewer window later in this tutorial.

Reviewing the Procedural Entities View

The procedural entities view, in the lower-left part of the Viewer window, displays a table with information about the diagnostics for each file in the project. The procedural entities view is also called the RTE (Run-Time Error) view. When you first open the results file from the verification of `example.adb`, the procedural entities view looks like this.



Procedural entities	!	✕	?	✓	Line	...	%	Deta
Example_Project	6	4	6	20			83	
ADA					1		0	
PKDATA					1		0	
RANDOM					1		0	
RUNTIME_ERROR	6	4	6	20	23		83	examp
RUNTIME_ERROR$SPEC					18		0	examp
SENSITIVITY					1		0	

The package `RUNTIME_ERROR` is red because its contains at least one definite run-time error. PolySpace software assigns each package the color of the most severe error found in that package. Note that the other entities displayed for the example project are simply black. This indicates that they contain specifications that were used for the analysis. The first column of the Procedural entities view names the procedural entity (the package or function). The following table describes some of the other columns in the procedural entities view.

Column Heading	Indicates
!	Number of red checks (for operations where an error always occurs)
✕	Number of gray checks (for unreachable code)
?	Number of orange checks (warnings for operations where an error might occur)

Column Heading	Indicates
✓	Number of green checks (for operations where an error never occurs)
%	Total number of red, green, and gray checks (an indication of the level of proof)

Tip If you see three dots in place of a heading, ..., resize the column until you see the heading. Resize the procedural entities view to see additional columns.

Note You can select which columns appear in the procedural entities view by editing the preferences. To learn how to add a **Reviewed** column, see "Making the Reviewed Column Visible" on page 4-13.

What you select in the procedural entities view determines what is displayed in the other views. In the following examples, you learn how to use the views and how they interact.

Reviewing Results in Expert Mode

In this section...

What Is Expert Mode?

In expert mode, you can see all checks from the verification in the PolySpace Viewer. You decide which checks to review and in what order to review them.

Switching to Expert Mode

If the Viewer is in expert mode, the switch mode button displays **Assistant**. If the Viewer is in assistant mode, the switch mode button displays **Expert**. To switch from assistant to expert mode:

- Click the Viewer mode button:

  The Viewer window toolbar displays buttons and menus specific to expert mode.

Reviewing Checks in Expert Mode

In this part of the tutorial, you learn how to use the Viewer window views to examine checks from a verification. This part of the tutorial covers:

- "Selecting a Check to Review" on page 4-10
- "Displaying the Calling Sequence" on page 4-11
- "Tracking Review Progress" on page 4-12

- “Making the Reviewed Column Visible” on page 4-13

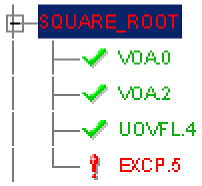
Selecting a Check to Review

In the procedural entities view, `RUNTIME_ERROR` is red, indicating that this package has at least one red check. To review a red check in `RUNTIME_ERROR`:

1 In the procedural entities section of the window, expand `RUNTIME_ERROR`.

2 Expand the red function `SQUARE_ROOT`.

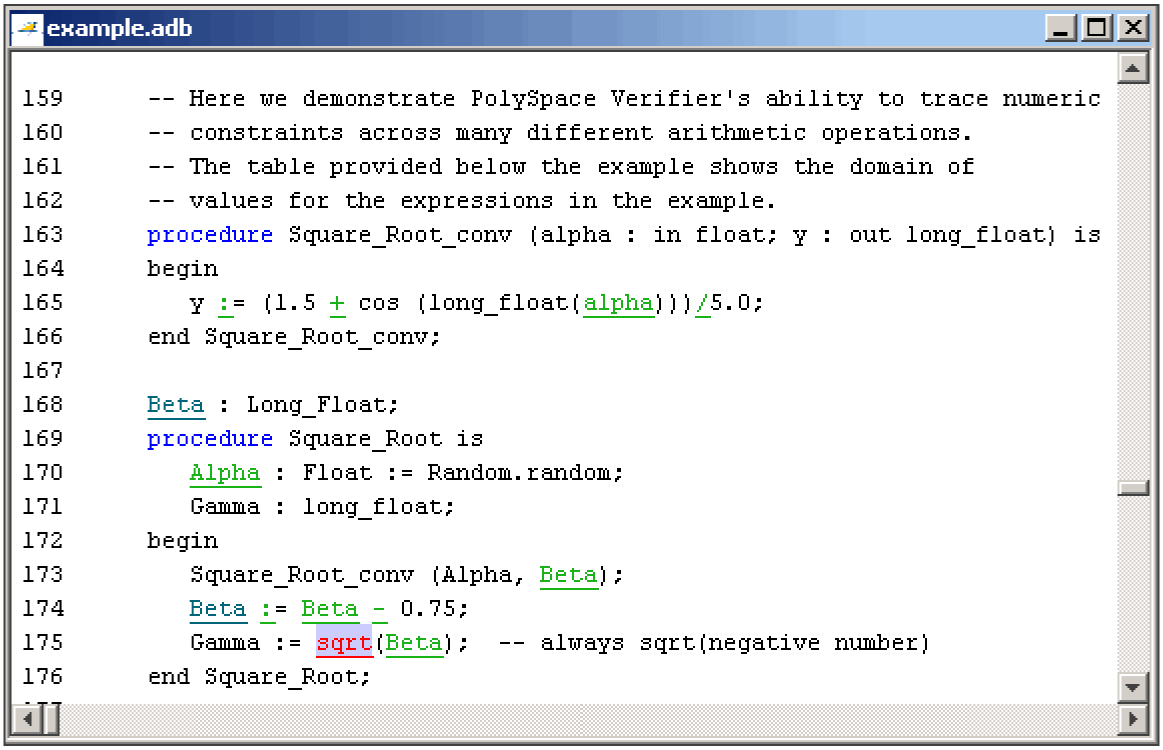
A color-coded list of the checks performed on `SQUARE_ROOT` appears:



Each item in the list of checks has an acronym that identifies the type of check and a number. For example, in `EXCP.5`, EXCP stands for Arithmetic Exception. For more information about different types of checks, see “Check Descriptions” in the *PolySpace Client/Server for Ada User’s Guide*.

3 Click on the red `EXCP.5`.

The source code view displays the section of source code where this error occurs.



```
159     -- Here we demonstrate PolySpace Verifier's ability to trace numeric
160     -- constraints across many different arithmetic operations.
161     -- The table provided below the example shows the domain of
162     -- values for the expressions in the example.
163     procedure Square_Root_conv (alpha : in float; y : out long_float) is
164     begin
165        y := (1.5 + cos (long_float(alpha)))/5.0;
166     end Square_Root_conv;
167
168     Beta : Long_Float;
169     procedure Square_Root is
170        Alpha : Float := Random.random;
171        Gamma : long_float;
172     begin
173        Square_Root_conv (Alpha, Beta);
174        Beta := Beta - 0.75;
175        Gamma := sqrt(Beta);  -- always sqrt(negative number)
176     end Square_Root;
```

4 At line 175 of the code, click on the red code.

An error message box appears indicating that when the local float variable `Gamma` is computed at line 175, the operation will cause a run-time error because the parameter passed to `sqrt` is always negative.

Displaying the Calling Sequence

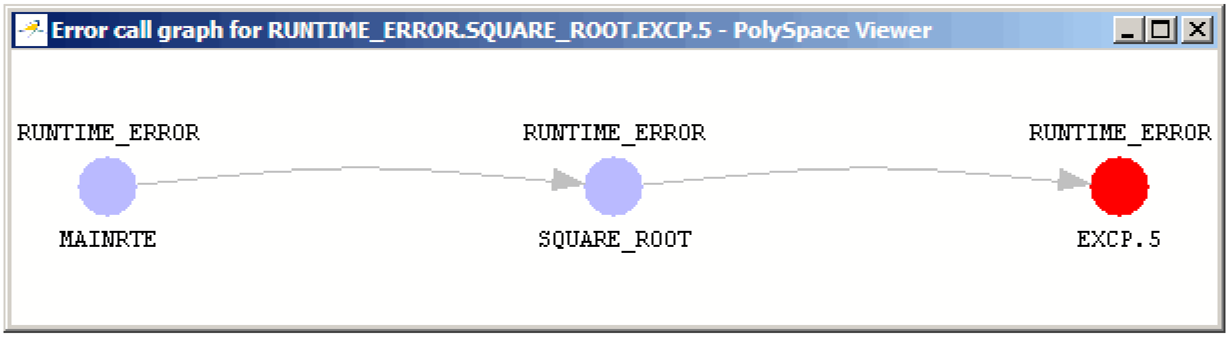
You can display the calling sequence that leads to the code associated with a check. To see the calling sequence for the red `EXCP.5` check in `SQUARE_ROOT`:

1 Expand `SQUARE_ROOT`.

2 Click on the red `EXCP.5`.

3 Click the call graph button in the toolbar.

A window displays the call graph.


Error call graph for RUNTIME_ERROR.SQUARE_ROOT.EXCP.5 - PolySpace Viewer
RUNTIME_ERROR
MAINRTE
RUNTIME_ERROR
SQUARE_ROOT
RUNTIME_ERROR
EXCP.5

The code associated with EXCP.5 is in SQUARE_ROOT. The function MAINRTE calls SQUARE_ROOT.

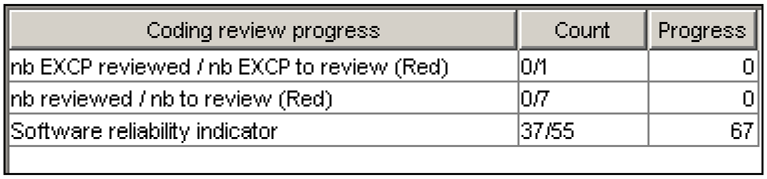
Tracking Review Progress

You can keep track of the checks that you have reviewed by marking them. To mark that you have reviewed the red EXCP.5 check in SQUARE_ROOT:

1 Expand SQUARE_ROOT.

2 Click the red EXCP.5.

A table with statistics about the review progress for that category and severity of error appear in the upper-left part of the window.

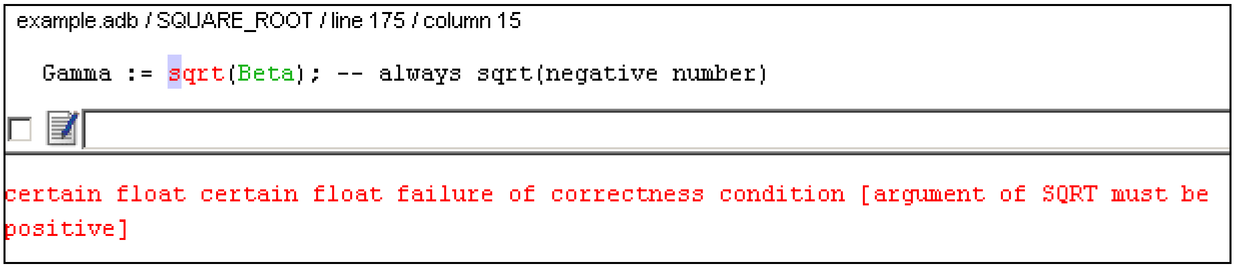


Coding review progress	Count	Progress
nb EXCP reviewed / nb EXCP to review (Red)	0/1	0
nb reviewed / nb to review (Red)	0/7	0
Software reliability indicator	37/55	67

The **Count** column displays a ratio and the **Progress** column displays the equivalent percentage. The first row displays the ratio of reviewed checks to the total number of checks that have the same color and category as the current check. In this example, it displays the ratio of reviewed red EXCP checks to total red EXCP errors in the project.

The second row displays the ratio of reviewed checks to total checks that have the same color as the current check. In this example, this is the ratio of red errors reviewed to total red errors in the project. The third row displays the ratio of the number of green checks to the total number of checks, providing an indicator of the reliability of the software.

Information about the current check (the red `EXCP.5`) appears in the upper-right part of the Viewer window.



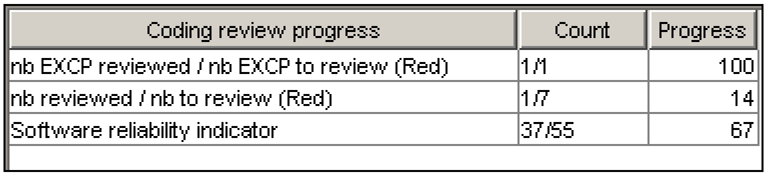
example.adb / SQUARE_ROOT / line 175 / column 15

```
  Gamma := sqrt(Beta); -- always sqrt(negative number)
```

```
certain float certain float failure of correctness condition [argument of SQRT must be
positive]
```

3 Select the check box to indicate that you have reviewed this check. You can enter a comment in the comment box.

The **Coding review progress** part of the window updates the ratios of errors reviewed to total errors.



Coding review progress	Count	Progress
nb EXCP reviewed / nb EXCP to review (Red)	1/1	100
nb reviewed / nb to review (Red)	1/7	14
Software reliability indicator	37/55	67

Making the Reviewed Column Visible

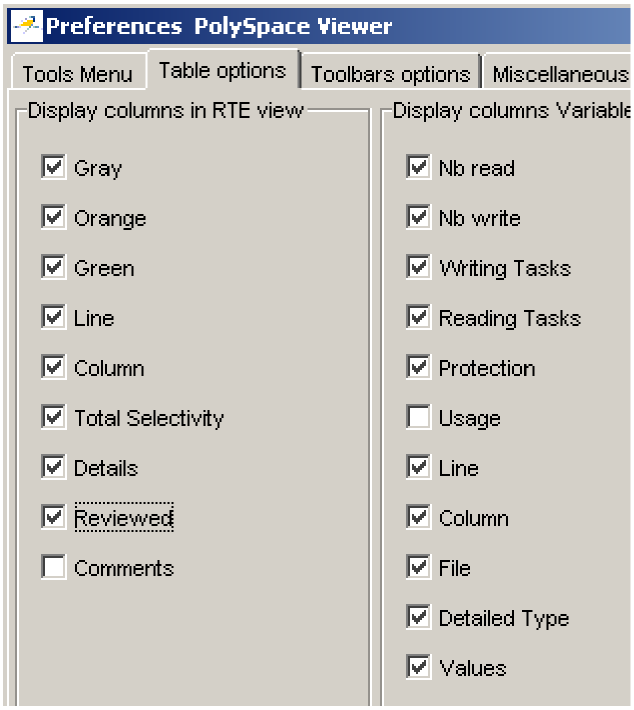
You can change the PolySpace Viewer preferences so that the procedural entities part of the window displays a **Reviewed** column.

1 Select **Edit > Preferences**.

2 Select the **Table options** tab.

3 Under **Display columns in RTE view**, select the **Reviewed** check box.

Now the **Table options** tab looks like this.


Preferences PolySpace Viewer

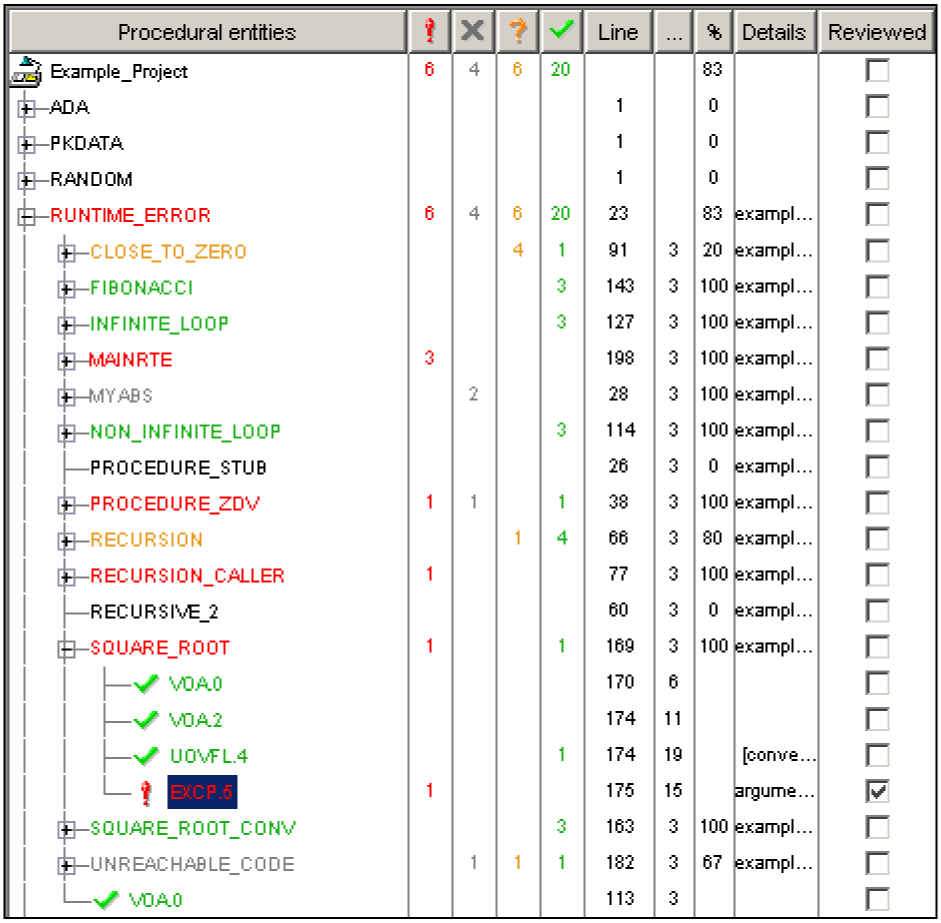
Tools Menu | Table options | Toolbars options | Miscellaneous

Display columns in RTE view
- [✓] Gray
- [✓] Orange
- [✓] Green
- [✓] Line
- [✓] Column
- [✓] Total Selectivity
- [✓] Details
- [✓] Reviewed
- [ ] Comments

Display columns Variable
- [✓] Nb read
- [✓] Nb write
- [✓] Writing Tasks
- [✓] Reading Tasks
- [✓] Protection
- [ ] Usage
- [✓] Line
- [✓] Column
- [✓] File
- [✓] Detailed Type
- [✓] Values

4 Click **OK** to apply the preference and close the dialog box.

A column of check boxes appears in the **Procedural entities** view.



Procedural entities	!	✗	?	✓	Line	...	%	Details	Reviewed
Example_Project	6	4	6	20			83		☐
ADA					1		0		☐
PKDATA					1		0		☐
RANDOM					1		0		☐
RUNTIME_ERROR	6	4	6	20	23		83	exampl...	☐
CLOSE_TO_ZERO			4	1	91	3	20	exampl...	☐
FIBONACCI				3	143	3	100	exampl...	☐
INFINITE_LOOP				3	127	3	100	exampl...	☐
MAINRTE	3				198	3	100	exampl...	☐
MYABS		2			28	3	100	exampl...	☐
NON_INFINITE_LOOP				3	114	3	100	exampl...	☐
PROCEDURE_STUB					26	3	0	exampl...	☐
PROCEDURE_ZDV	1	1		1	38	3	100	exampl...	☐
RECURSION			1	4	66	3	80	exampl...	☐
RECURSION_CALLER	1				77	3	100	exampl...	☐
RECURSIVE_2					60	3	0	exampl...	☐
SQUARE_ROOT	1			1	169	3	100	exampl...	☐
VOA.0					170	6			☐
VOA.2					174	11			☐
UOVFL.4				1	174	19		[conve...	☐
EXCP.5	1				175	15		argume...	☑
SQUARE_ROOT_CONV				3	163	3	100	exampl...	☐
UNREACHABLE_CODE		1	1	1	182	3	67	exampl...	☐
VOA.0					113	3			☐

The check box for `EXCP.5` in `SQUARE_ROOT` is selected because you selected the check box for this diagnostic in the current check view (upper- right part of window).

Tip If you do not see this column, resize **Procedural entities** so that you see the column. Resize the column to see the **Reviewed** label.

Note Selecting a check box in the **Reviewed** column automatically:

- Selects the check box for that check in the current check view (upper-right part of the window).
- Updates the counts in the coding review progress view (upper-left part of the window).

Reviewing Additional Examples of Checks

In this part of the tutorial, you learn about other types and categories of errors by reviewing the following checks in `example.adb`:

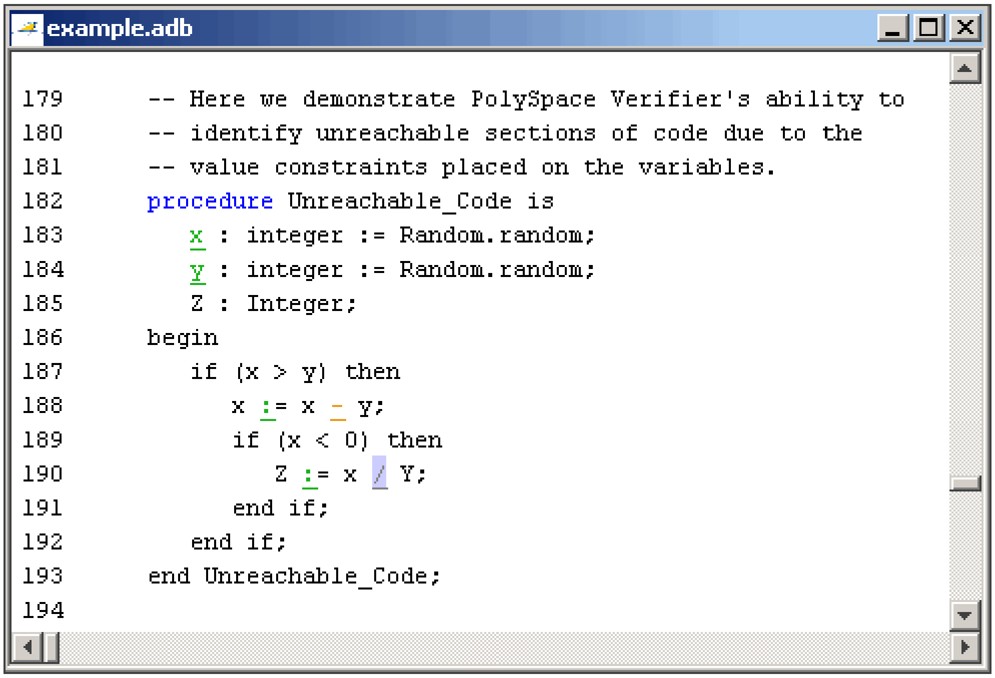
- "Example: Unreachable Code" on page 4-16
- "Example: A Function with No Errors" on page 4-17
- "Example: Division by Zero" on page 4-18

Example: Unreachable Code

Unreachable code is code that never executes. PolySpace software displays unreachable code in gray. In the following steps, you will look at an example of unreachable code.

1 In **Procedural Entities**, expand `UNREACHABLE_CODE` and click on the gray `ZDV.5`.

  The source code for this function displays in the source code view.



```
179      -- Here we demonstrate PolySpace Verifier's ability to
180      -- identify unreachable sections of code due to the
181      -- value constraints placed on the variables.
182      procedure Unreachable_Code is
183         x : integer := Random.random;
184         y : integer := Random.random;
185         Z : Integer;
186      begin
187         if (x > y) then
188            x := x - y;
189            if (x < 0) then
190               Z := x / Y;
191            end if;
192         end if;
193      end Unreachable_Code;
194
```

2 Examine the source code.

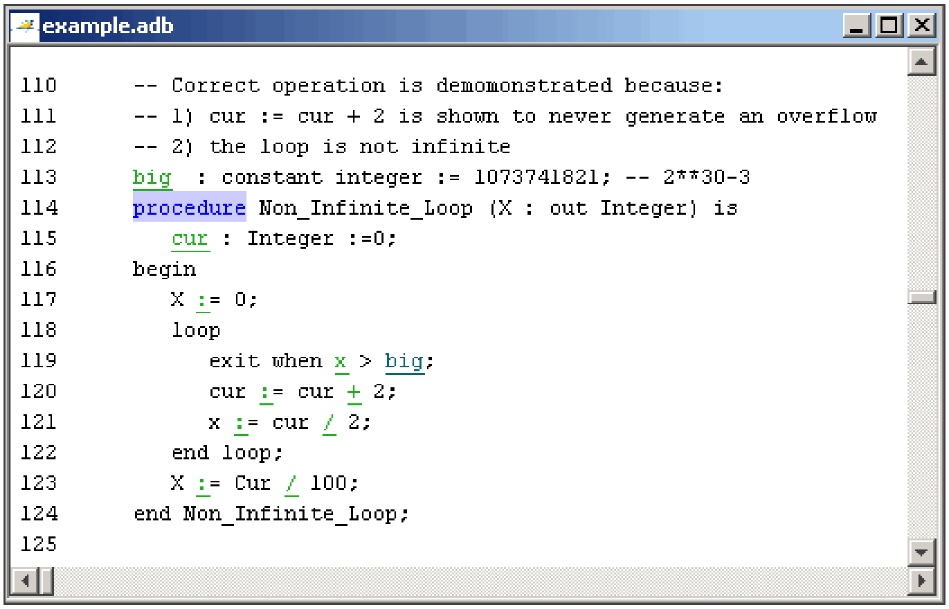
At line 190, the code `Z := x / Y;` is never reached because the condition `x < 0` is always false.

Example: A Function with No Errors

In the following example, PolySpace software determines, in code with a large number of iterations, that a loop terminates and a variable does not overflow:

1 In **Procedural entities**, click on the green `NON_INFINITE_LOOP` function.

The source code for this function is displayed in the source code view.



```
110      -- Correct operation is demomonstrated because:
111      -- 1) cur := cur + 2 is shown to never generate an overflow
112      -- 2) the loop is not infinite
113      big  : constant integer := 1073741821; -- 2**30-3
114      procedure Non_Infinite_Loop (X : out Integer) is
115         cur : Integer :=0;
116      begin
117         X := 0;
118         loop
119            exit when x > big;
120            cur := cur + 2;
121            x := cur / 2;
122         end loop;
123         X := Cur / 100;
124      end Non_Infinite_Loop;
125
```

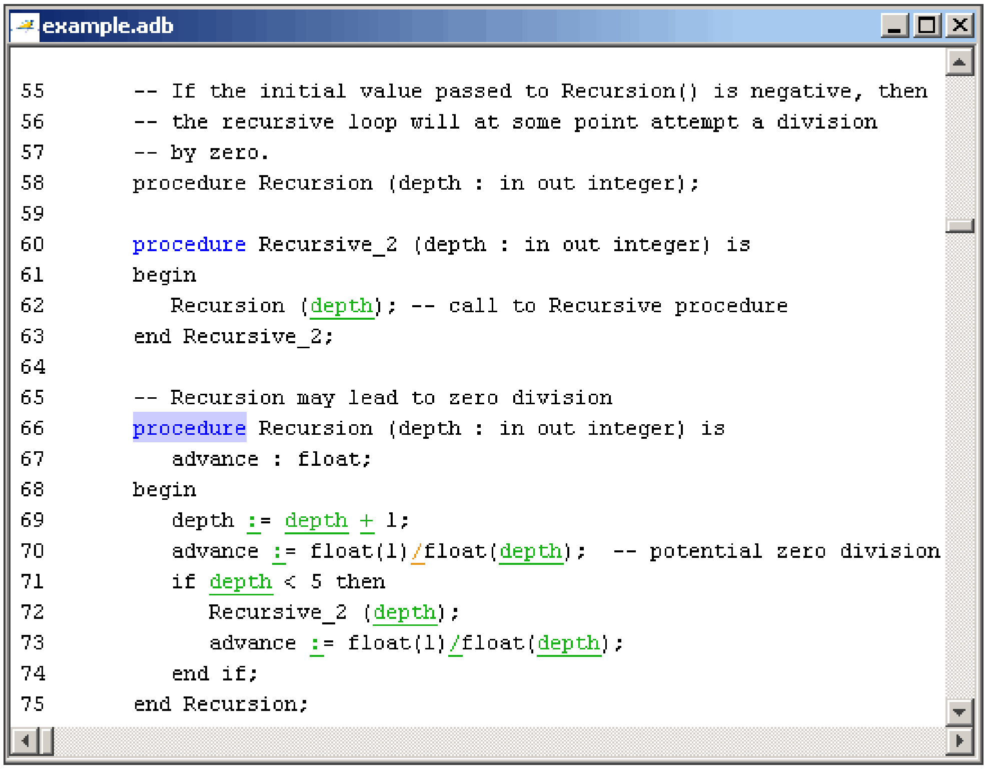
2 Examine the source code. The variable `cur` never overflows because the loop at line 117 terminates before `cur` can overflow.

Example: Division by Zero

In the following example, PolySpace software detects a potential division by zero:

1 In **Procedural entities**, expand `RECURSION`.

The source code for this function is displayed in the source code view.



```
55      -- If the initial value passed to Recursion() is negative, then
56      -- the recursive loop will at some point attempt a division
57      -- by zero.
58      procedure Recursion (depth : in out integer);
59
60      procedure Recursive_2 (depth : in out integer) is
61      begin
62         Recursion (depth); -- call to Recursive procedure
63      end Recursive_2;
64
65      -- Recursion may lead to zero division
66      procedure Recursion (depth : in out integer) is
67         advance : float;
68      begin
69         depth := depth + 1;
70         advance := float(1)/float(depth);  -- potential zero division
71         if depth < 5 then
72            Recursive_2 (depth);
73            advance := float(1)/float(depth);
74         end if;
75      end Recursion;
```

2 Examine the `RECURSION` function.

When `RECURSION` is called with `depth` less than zero, the code at line `70` will result in division by zero. The orange color indicates that this is a potential error (depending on the value of `depth`).

Filtering the Types of Checks That You See

You can filter the checks that you see in the Viewer so that you can focus on certain types of checks. PolySpace software provides three predefined composite filters, a custom composite filter, and several individual filters. You learn about filters in the following sections:

- "Using Composite Filters" on page 4-20

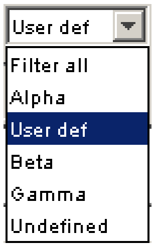
- "Using the Custom Filter" on page 4-21
- "Using Individual Filters" on page 4-23

Using Composite Filters

Composite filters combine individual filters, allowing you to display or hide groups of checks.

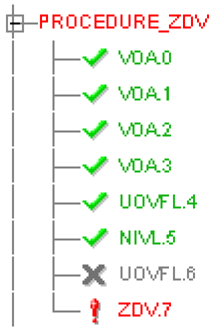
Use this filter...	To...
Alpha	Display all checks
Beta	Hide NIV, NIVL, NIP, Scalar OVFL, and Float OVFL checks
Gamma	Display red and gray checks
User def	Hide checks as defined in a custom filter that you can modify

The default filter is `User def`. You learn more about the `User def` filter in "Using the Custom Filter" on page 4-21. You can select a composite filter from the filter menu.



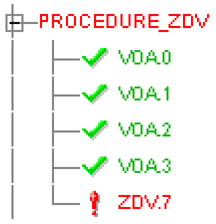
To learn how the composite filters affect the display of checks:

1 Expand the function `PROCEDURE_ZDV` in **Procedural Entities**. Select `Alpha` from the filter menu to display all the checks for `PROCEDURE_ZDV`.



PROCEDURE_ZDV has eight checks: six green, one gray, and one red.

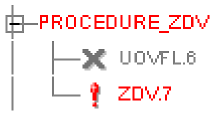
2 Select Beta from the filter menu to hide the NIV local, SCAL OVFL, NIV other, and FLOAT OVFL checks.



Now, only five checks are visible: four VOA, and one ZDV.

3 Select Alpha to display all checks again.

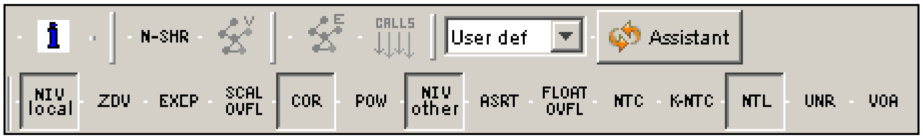
4 Select Gamma to display only the red and gray checks.



Now, only two checks are visible: the gray UOVFL and the red ZDV.

Using the Custom Filter

The custom filter is a composite filter that you define. It appears on the composite filter menu as User def and is the default composite filter. By default, the custom filter hides the NIV local, COR, NIV other, and NTL checks as shown in the following figure.

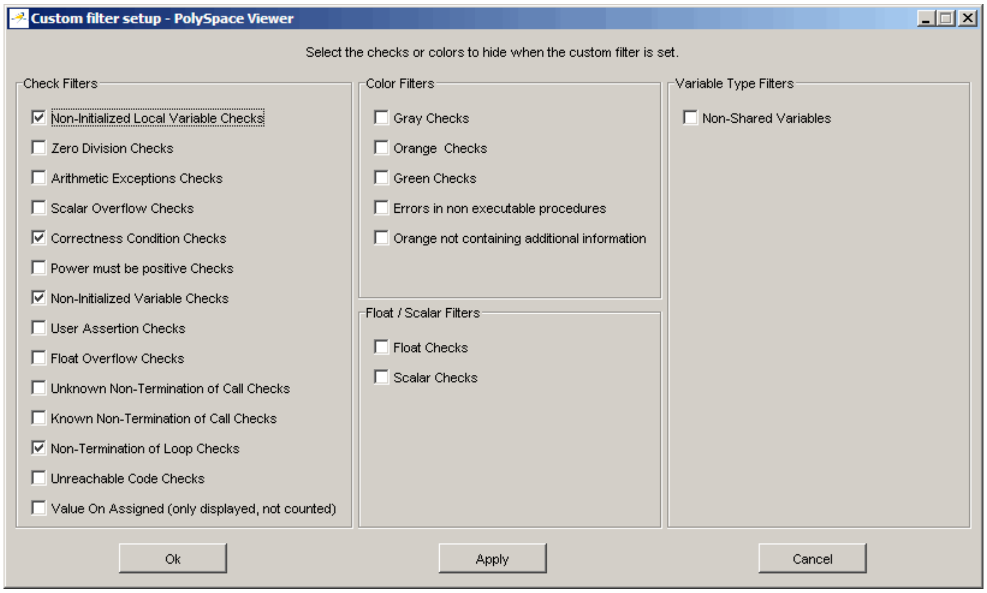


To modify the custom filter:

1 Select `User def` from the composite filters menu.

2 Select **Edit > Custom filters**.

The **Custom filter setup** dialog box appears.


Custom filter setup - PolySpace Viewer

Select the checks or colors to hide when the custom filter is set.

Check Filters
- [✓] Non-Initialized Local Variable Checks
- [ ] Zero Division Checks
- [ ] Arithmetic Exceptions Checks
- [ ] Scalar Overflow Checks
- [✓] Correctness Condition Checks
- [ ] Power must be positive Checks
- [✓] Non-Initialized Variable Checks
- [ ] User Assertion Checks
- [ ] Float Overflow Checks
- [ ] Unknown Non-Termination of Call Checks
- [ ] Known Non-Termination of Call Checks
- [✓] Non-Termination of Loop Checks
- [ ] Unreachable Code Checks
- [ ] Value On Assigned (only displayed, not counted)

Color Filters
- [ ] Gray Checks
- [ ] Orange Checks
- [ ] Green Checks
- [ ] Errors in non executable procedures
- [ ] Orange not containing additional information

Float / Scalar Filters
- [ ] Float Checks
- [ ] Scalar Checks

Variable Type Filters
- [ ] Non-Shared Variables

3 Clear the filters for the checks that you want to display. For example, if you clear the **Non-Initialized Local Variables Checks** box, these checks display.

Note You do not have to change any of the selections for this tutorial.

4 Select the filters for the checks that you do not want to display.

5 Click **OK** to apply the changes and close the dialog box.

PolySpace software saves the custom filter definition in the Viewer preferences.

Using Individual Filters

You can use an individual filter to display or hide a given check category. When a filter is enabled, that check category is not displayed. For example, when the VOA filter is enabled, VOA checks are not displayed. When the VOA filter is disabled, VOA checks are displayed. You can also filter by check color. To enable or disable an individual filter, click the toggle button for that filter on the toolbar.

Tip When you mouse over a filter button, a tooltip tells you which filter the button is for and whether the filter is currently enabled or disabled.

To learn how an individual filter affects the display of checks:

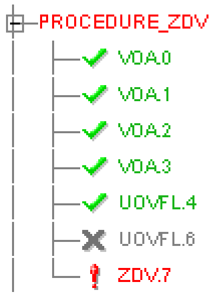
1 Expand `PROCEDURE_ZDV`.

2 Select `Alpha` from the composite filters menu to display all checks.

3 Click the **NIV local** filter button

NIV local

to hide the NIVL checks for `PROCEDURE_ZDV`.

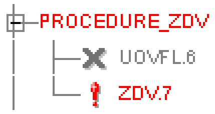

PROCEDURE_ZDV
VOA.0
VOA.1
VOA.2
VOA.3
UOVFL.4
UOVFL.6
ZDV.7

4 Click the **NIV local** filter button again to display the NIVL checks.

5 Now click the green checks filter button

✓

to hide the green checks.


PROCEDURE_ZDV
UOVFL.6
ZDV.7

Note When you filter a check category, some red checks within that category are still displayed. For example, if you filter ZDV checks, `ZDV.7` is still displayed under `PROCEDURE_ZDV`.

Reviewing Results in Assistant Mode

In this section...

What Is Assistant Mode?

In assistant mode, PolySpace software chooses the checks for you to review and the order in which you review them. PolySpace software presents checks to you in this order:

1. All red checks
2. All blocks of gray checks (the first check in each unreachable function)
3. Orange checks according to the selected methodology and criterion level

You will learn about methodologies and criterion levels in "Selecting the Methodology and Criterion Level" on page 4-26.

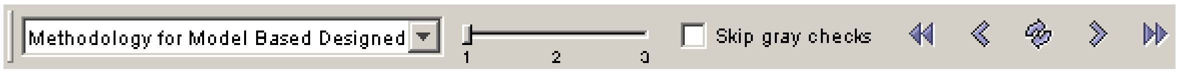
Switching to Assistant Mode

If the Viewer is in assistant mode, the switch mode button displays **Expert**. If the Viewer is in expert mode, the switch mode button displays **Assistant**. To switch from expert mode to assistant mode:

- Click the Viewer's switch mode button

.

  The Viewer window toolbar displays controls specific to assistant mode.



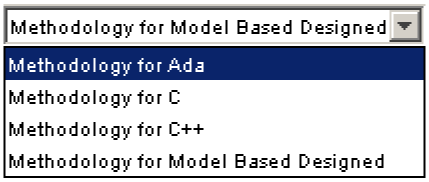
The controls for assistant mode include:

- A menu for selecting the review methodology for orange checks
- A slider for selecting the criterion level within that methodology
- A check box for skipping gray checks
- Arrows for navigating through the reviews

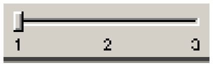
Selecting the Methodology and Criterion Level

A methodology is a named configuration that defines the number of orange checks, by category, that you review in assistant mode. Each methodology has three criterion levels. Each level specifies the number of orange checks for a given category. The levels correspond to different development phases that have different review requirements. To select the methodology and level for this tutorial:

1 Select **Methodology for Ada** from the methodology menu.


Methodology for Model Based Designed
Methodology for Ada
Methodology for C
Methodology for C++
Methodology for Model Based Designed

2 If the level slider is not already at 1, move the slider to level 1.



Exploring Methodology for Ada

In this part of the tutorial, you examine the configuration for **Methodology for Ada**. To begin:

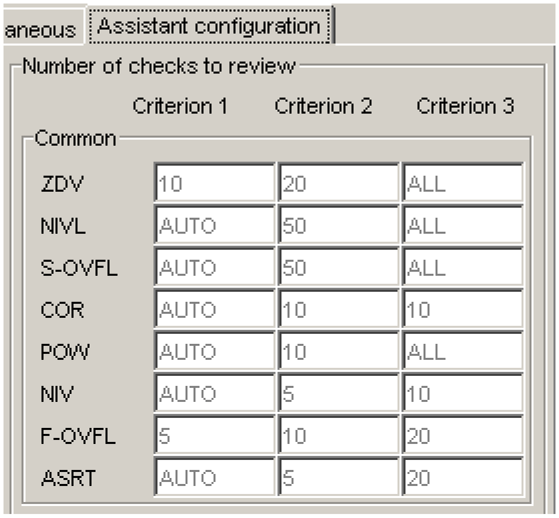
1 Select **Edit > Preferences**.

The **Preferences PolySpace Viewer** dialog box appears.

2 Select the **Assistant configuration** tab.

The configuration for Methodology for Ada appears.

On the right side of the dialog box, a table shows the number of orange checks that you review for a given criterion and check category.



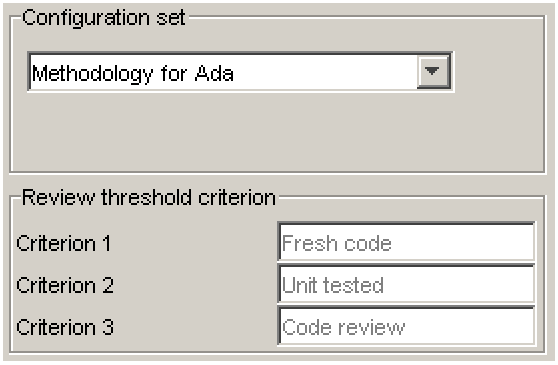
Number of checks to review

Common

	Criterion 1	Criterion 2	Criterion 3
ZDV	10	20	ALL
NIVL	AUTO	50	ALL
S-OVFL	AUTO	50	ALL
COR	AUTO	10	10
POW	AUTO	10	ALL
NIV	AUTO	5	10
F-OVFL	5	10	20
ASRT	AUTO	5	20

For example, the table specifies that you review ten orange ZDV checks when you select criterion 1. The number of checks increases as you move from criterion 1 to criterion 3, reflecting the changing review requirements as you move through the development process.

In the lower-left part of the dialog box, the section **Review threshold criterion** contains text that appears in the tooltip for the criterion slider on the Viewer toolbar (only in assistant mode).



Methodology for Ada

Criterion 1	Fresh code
Criterion 2	Unit tested
Criterion 3	Code review

For the configuration Methodology for Ada, the criterion names are:

Criterion	Name in the Tooltip
1	Fresh code
2	Unit tested
3	Code review

These names correspond to phases of the development process.

3 Click **OK** to close the dialog box.

Reviewing Checks

In assistant mode, you review checks in the order in which PolySpace software presents them:

1 All reds

2 All blocks of gray checks (the first check in each unreachable function)

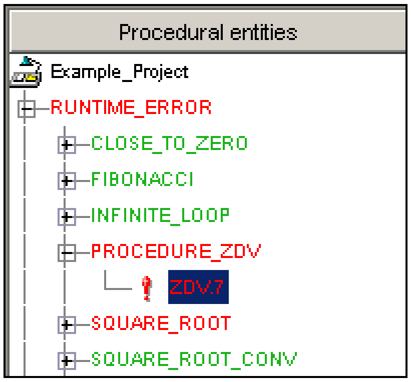
Note You can skip gray checks by selecting the **Skip gray checks** check box in the toolbar.

3 Orange checks according to the selected methodology and criterion level

Earlier in this tutorial, you selected Methodology for Ada, criterion 1. In this part of the tutorial, you continue to review the checks for `example.adb` using this methodology and criterion level. To navigate through these checks:

1 In the procedural entities view (lower left), `PROCEDURE_ZDV` is expanded and `ZDV.7` is displayed as the current check.

If the Viewer is displaying the message "No check currently selected" in the upper-right portion of the window, then you will need to click the forward arrow ⟩ to go to the first check.



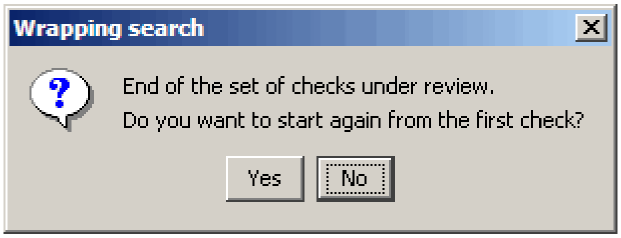
The source code view (lower right) displays the source for this check and the current check view (upper right) displays information about this check.

Note You can display the calling sequence and track review progress as you did in "Reviewing Results in Expert Mode" on page 4-9.

---

2 Continue to click the forward arrow until you have gone through all of the checks.

After the last check, a dialog box appears asking if you want to start again from the first check.



Wrapping search

End of the set of checks under review.
Do you want to start again from the first check?

3 Click **No**.

Defining a Custom Methodology

You cannot change the predefined methodologies, such as Methodology for Ada, but you can define your own methodology. In this part of the tutorial, you learn how to create and use your own methodology.

To define your custom methodology:

1 Select **Edit > Preferences**.

   The **Preferences PolySpace Viewer** dialog box appears.

2 Select the **Assistant configuration** tab.

3 Select **Add a set** from the menu in **Configuration set**.

4 In the **Create a new set** dialog box, enter `My methodology` for the name and click **Enter** to close the dialog box.

5 Under the **Criterion 1** column, enter the number 1 next to **ZDV** . This tells PolySpace software to select up to one orange ZDV for review. PolySpace will not select any other orange checks for review because you are leaving all of the other fields blank.

   This does not affect the red and gray checks: the software will still present all red checks and the first check in each unreachable function for review.

6 Click **OK** to save the methodology and close the dialog box.

To use `My methodology`:

1 Select `My methodology` from the methodology menu.

2 If the level slider is not already at 1, move the slider to level 1.

3 Click the forward arrow to review the checks.

With this methodology at criterion 1, the only orange check you review is the orange ZDV.5 in `RECURSION`.

4 End PolySpace Viewer by selecting **File > Quit**.

Generating Reports of Verification Results

Generating a Report of the example.adb Verification

You can generate a Microsoft Excel® report of the verification results. To generate an Excel report of the verification results for the package `example.adb`:

1 Navigate to `polypace_project\results\PolySpace-Doc`.

   The directory should have the following files:

   ```
   Example_Project_Call_Tree.txt
   Example_Project_RTE_View.txt
   Example_Project_Variable_View.txt
   Example_Project-NON-SCALAR-TABLE-APPENDIX.ps
   PolySpace_Macros.xls
   ```

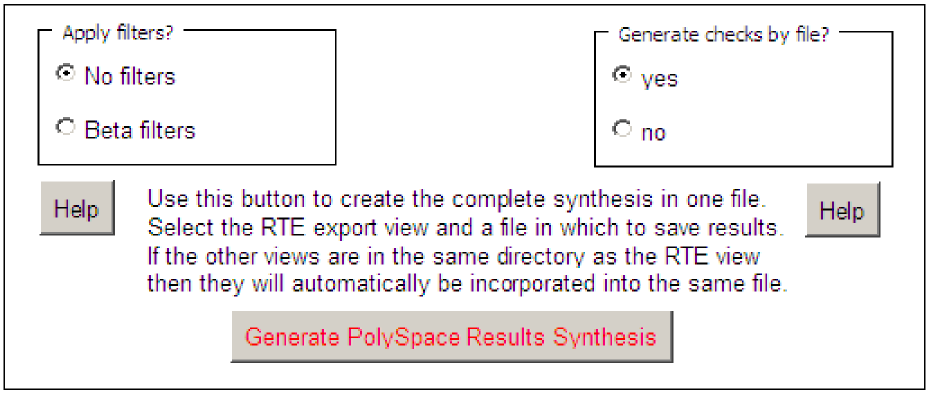
   The first three files correspond to the call tree, RTE, and variable views in the PolySpace Viewer window. For more information about the Viewer window, see “Exploring the Viewer Window” on page 4-5.

2 Open the macros file `PolySpace_Macros.xls`.

   A security warning dialog appears.

3 Click **Enable Macros**.

   A spreadsheet appears. The top part of the spreadsheet looks like this.


Apply filters?
[✓ No filters]
Beta filters

Generate checks by file?
[✓ yes]
no

Use this button to create the complete synthesis in one file. Select the RTE export view and a file in which to save results. If the other views are in the same directory as the RTE view then they will automatically be incorporated into the same file.

4 In the top half of the spreadsheet, in **Apply filters?**, select **No filters**.

5 In **Generate checks by file?**, select the **yes**.

6 Click **Generate PolySpace Results Synthesis**.

   The synthesis report combines the RTE, call tree, and variables views into one report.

   The **Select a RTE View text file** dialog box appears.

7 In **Look in**, navigate to `polypace_project\results\PolySpace-Doc`.

8 Select `Example_Project_RTE_View.txt`.

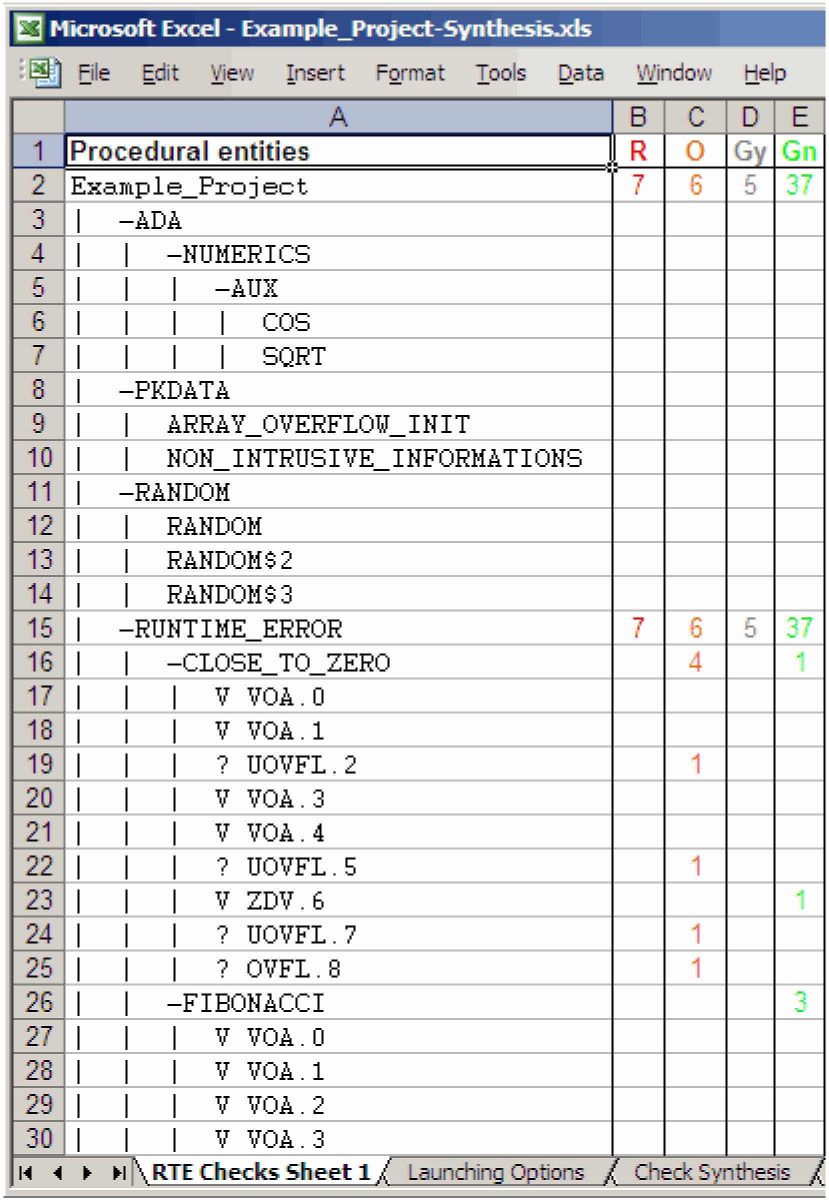
9 Click **Open** to close the dialog box.

   The **Where should I save the analysis file?** dialog box appears.

10 Keep the default file name `Example_Project-Synthesis` and file type `Microsoft Excel Workbook(*.xls)`

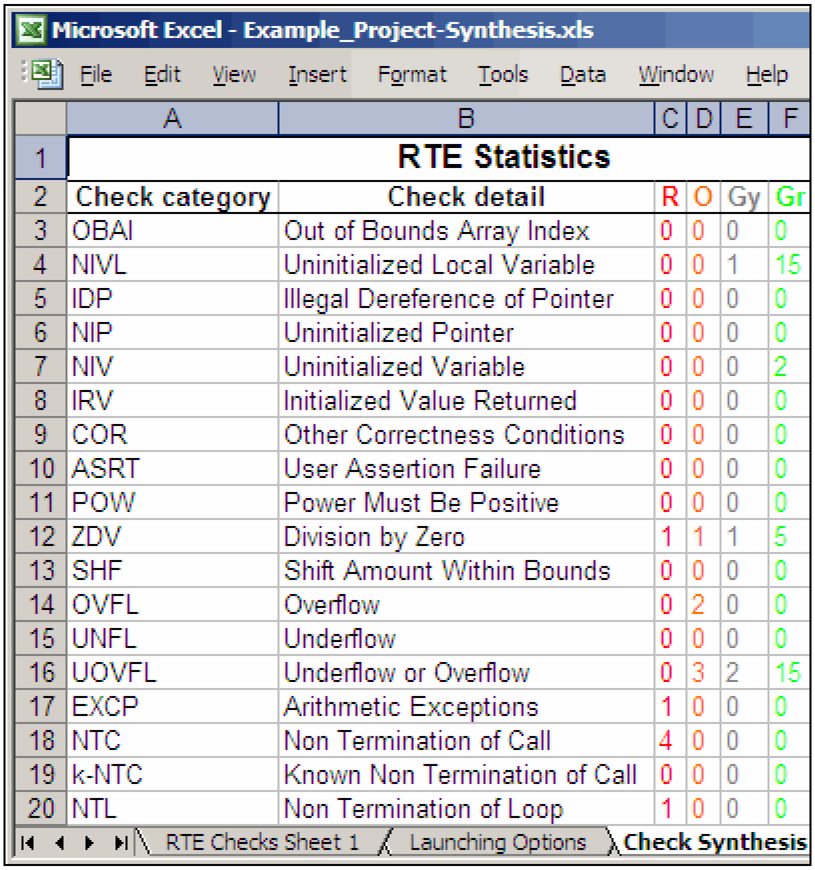
11 Click **Save** to close the dialog box and start the report generation.

   Microsoft Excel opens with the spreadsheet that you generated. This spreadsheet has several worksheets:


Microsoft Excel - Example_Project-Synthesis.xls
File Edit View Insert Format Tools Data Window Help
RTE Checks Sheet 1
Launching Options
Check Synthesis

	A	B	C	D	E
1	Procedural entities	R	O	Gy	Gn
2	Example_Project	7	6	5	37
3	| -ADA				
4	| | -NUMERICS				
5	| | | -AUX				
6	| | | | COS				
7	| | | | SQRT				
8	| -PKDATA				
9	| | ARRAY_OVERFLOW_INIT				
10	| | NON_INTRUSIVE_INFORMATIONS				
11	| -RANDOM				
12	| | RANDOM				
13	| | RANDOM$2				
14	| | RANDOM$3				
15	| -RUNTIME_ERROR	7	6	5	37
16	| | -CLOSE_TO_ZERO		4		1
17	| | | V VOA.0				
18	| | | V VOA.1				
19	| | | ? UOVFL.2		1		
20	| | | V VOA.3				
21	| | | V VOA.4				
22	| | | ? UOVFL.5		1		
23	| | | V ZDV.6				1
24	| | | ? UOVFL.7		1		
25	| | | ? OVFL.8		1		
26	| | -FIBONACCI				3
27	| | | V VOA.0				
28	| | | V VOA.1				
29	| | | V VOA.2				
30	| | | V VOA.3				

12 Select the **Check Synthesis** tab to view the worksheet showing statistics by check category:


Microsoft Excel - Example_Project-Synthesis.xls
File Edit View Insert Format Tools Data Window Help
RTE Checks Sheet 1
Launching Options
Check Synthesis

RTE Statistics

Check category	Check detail	R	O	Gy	Gr
OBAI	Out of Bounds Array Index	0	0	0	0
NIVL	Uninitialized Local Variable	0	0	1	15
IDP	Illegal Dereference of Pointer	0	0	0	0
NIP	Uninitialized Pointer	0	0	0	0
NIV	Uninitialized Variable	0	0	0	2
IRV	Initialized Value Returned	0	0	0	0
COR	Other Correctness Conditions	0	0	0	0
ASRT	User Assertion Failure	0	0	0	0
POW	Power Must Be Positive	0	0	0	0
ZDV	Division by Zero	1	1	1	5
SHF	Shift Amount Within Bounds	0	0	0	0
OVFL	Overflow	0	2	0	0
UNFL	Underflow	0	0	0	0
UOVFL	Underflow or Overflow	0	3	2	15
EXCP	Arithmetic Exceptions	1	0	0	0
NTC	Non Termination of Call	4	0	0	0
k-NTC	Known Non Termination of Call	0	0	0	0
NTL	Non Termination of Loop	1	0	0	0

Index