

Release Notes for Polyspace[®] Products for Ada

How to Contact MathWorks



www.mathworks.com
comp.soft-sys.matlab
www.mathworks.com/contact_TS.html

Web
Newsgroup
Technical Support



suggest@mathworks.com
bugs@mathworks.com
doc@mathworks.com
service@mathworks.com
info@mathworks.com

Product enhancement suggestions
Bug reports
Documentation error reports
Order status, license renewals, passcodes
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.

Release Notes for Polyspace® Products for Ada

© COPYRIGHT 2013 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.

R2013a

Polyspace Client for Ada Product	2
Modified Polyspace installer	3
Verification uses native binary file	5
Exact representation of floating point numbers	6
Changes to verification results	7
Changes to analysis options	8
Options removed	9
Polyspace Server for Ada Product	10
Improved Polyspace Metrics security with HTTPS	11

R2012b

Polyspace Client for Ada Product	14
Review of verification results improvement	15
Reorganized Configuration pane	17
Polyspace plug-in for Eclipse with GNATbench	18
Report content filtering	19
Parent folder for verification results	20
Support for relative paths	21
Intermediate verification level support	22
Automatic import of comments and justifications	23
Storage of temporary files	24
Removal of Polyspace in One Click	25
Changes to verification results	26
Changes to analysis options	27
Options removed	32
Polyspace Server for Ada Product	33
Password-protected access to projects in Polyspace Metrics	34

Polyspace Client for Ada Product	36
Compilation Environment Templates	37
Suppression of NTC, NTL and UNR Checks Caused by Red Checks	39
Redefinition of Successful Verification	40
Polyspace Report Generator Enhancements	41
Polyspace In One Click (POC) Enhancement	42
Header Files Without Run-Time Checks	43
Improved Access to Polyspace Demos	44
Changes to Verification Results	45
Changes to Analysis Options	46
Options Removed	47
Polyspace Server for Ada Product	48
Enhanced Polyspace Metrics Project Index	49
Redefinition of Successful Verification	50

Polyspace Client for Ada Product	52
Review Orange Checks that are Potential Run-Time Errors	53
No Gray Checks in Unreachable Code	54
Global Variable Range Information	55
Read and Write Access in Dead Code	56
Run All Verifications in Project	57
Green NIV check for Unchecked_Conversion Function ...	58
Polyspace Server for Ada Product	59
Running Multiple Verifications Simultaneously	60
Polyspace Metrics	61

Polyspace Client for Ada Product	64
Code Metrics	65
Saving Polyspace Metrics Review	66
Support for Rational and Aonix Compilers	67
Multi-Core Support	68
Generated Main with Explicit Tasks and Accept Statements	69
Enhancements in Run-Time Checks Perspective	70
UOVFL and UNFL Checks Removed	71
NIV Checks for Universal Constants	72
Variable Range Inconsistency between Variable Access Pane and Tooltips	73
Verification Time Limit	74
Automatic Addition of Specifications for Selected Source Files	75
Stubbed Tasks	76
Scaling Issue for Large Applications with Nested Structures/Arrays	77
Product Name Change in Files and Folders	78
License Manager Support	79
Changes to Verification Results	80
Changes to Analysis Options	83
Polyspace Server for Ada Product	84
Code Metrics	85
Saving Polyspace Metrics Review	86
Multi-Core Support	87
Generated Main with Explicit Tasks and Accept Statements	88
Automatic Comment Import for Server Verifications	89
License Manager Support	90

Polyspace Client for Ada Product	92
Polyspace Graphical User Interface	93

Data Range Specifications	96
Extended Support for Tasks	97
Preprocessor Macros for Compilation	98
New Options to Classify Run-Time Checks	99
Permissiveness on File and Folder Names	101
Default Target Processor	102
Operating System Support	103
Polyspace Server for Ada Product	104
Polyspace Metrics Web Interface	105
Automatic Verification	106
Operating System Support	107

R2010a

Polyspace Client for Ada Product	110
License Activation	111
Source Code Comment Support	113
Eclipse Integration	114
Operating System Support	115
Polyspace Server for Ada Product	116
License Activation	117
Queue Manager Interface	119
Operating System Support	120

R2009b

Polyspace Client for Ada Product	122
Report Generator	123
Main Generator Enhancements	125
Global Data Graphs	130
Unit-by-unit Verification	131
Operating System Support	132

Polyspace Server for Ada Product	133
Operating System Support	134

R2009a

Polyspace Client for Ada Product	136
Character Encoding Options	137
Operating System Support	138
 Polyspace Server for Ada Product	 139
Operating System Support	140

R2008b

Polyspace Client for Ada Product	142
Operating System Support	143
 Polyspace Server for Ada Product	 144
Operating System Support	145

R2008a

Polyspace Client for Ada Product	148
Removed Cygwin Software Dependency for Windows	
Platforms	149
Enhanced Installer	151
Viewer Improvements	152
Enhanced Compilation Checks	153
One-Click Enhancements	154
Operating System Support	155

Polyspace Server for Ada Product	156
Removed Cygwin Software Dependency for Windows	
Platforms	157
Enhanced Installer	159
Operating System Support	160

R2013a

Version: 6.5
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Modified Polyspace installer

In R2013a:

- The Polyspace® installer does not create the *Polyspace_Common* folder.
- The Polyspace product does not depend on environment variables, that is, the installer does not create POLYSPACE_* variables.
- The Remote Launcher Manager does not open automatically during the installation process. In addition, if the service or daemon was running before the installation of the new product, the service or daemon is not automatically restarted. You must manually open the Remote Launcher Manager, configure your Polyspace Server, and then start the service or daemon.
- You must manually install Polyspace plug-in (or add-in) files. The Polyspace installer does not automatically activate these files. For example, before you run a verification from the Eclipse™ IDE, you must manually install the Polyspace plug-in file.

The following table provides information about new locations for files and folders.

Binary files	Location in previous release	Location in R2013a
Polyspace verification environment	<i>Polyspace_Install</i> \PVE\Polyspace.exe <i>Polyspace_Install</i> /PVE/bin/polyspace	<i>Polyspace_Install</i> \polyspace \bin\polyspace[.exe]
Polyspace for Ada	<i>Polyspace_Install</i> \Verifier\wbin <i>Polyspace_Install</i> /Verifier/bin	<i>Polyspace_Install</i> \polyspace \bin
Spooler	<i>Polyspace_Common</i> \RemoteLauncher \wbin\PSQueueSpooler.exe <i>Polyspace_Common</i> /RemoteLauncher /bin/polyspace-spooler	<i>Polyspace_Install</i> \polyspace \bin\polyspace-spooler[.exe]
Polyspace plug-in for Eclipse	<i>Polyspace_Common</i> \PolyspaceForEclipse	<i>Polyspace_Install</i> \polyspace \plugin\eclipse

Binary files	Location in previous release	Location in R2013a
Polyspace plug-in for IBM® Rational® Rhapsody®	<i>Polyspace_Common</i> \PolyspaceUMLLink	<i>Polyspace_Install</i> \polyspace\plugin\rhapsody
Remote Launcher	<i>Polyspace_Common</i> \RemoteLauncher	<i>Polyspace_Install</i> \polyspace\bin
Report Generator	<i>Polyspace_Common</i> \ReportGenerator	<i>Polyspace_Install</i> \polyspace\bin\polyspace-report-generator

For more information, see:

- “Polyspace Software Administration”
- “Install Polyspace Plug-In for Eclipse IDE”

Verification uses native binary file

In R2013a, the Polyspace verification is run using the native binary file for your computer architecture. Previously, you specified a 32-bit or 64-bit verification through the option `-machine-architecture option_value`. Now, if you specify `-machine-architecture`, the software ignores the option and generates the following warning:

```
Option -machine-architecture option_value is obsolete.  
Verification run using native binary file for your  
computer architecture.
```

Previously, the default option value (auto) specified 32-bit verification. Now, on a Linux® system, the Polyspace verification is 64-bit. You may observe an increase in verification time compared to previous releases. The increase depends on the application being verified and the architecture of your machine.

Note For Linux systems, only the Polyspace 64-bit Client and Server products are supported.

Exact representation of floating point numbers

Polyspace uses the exact value of a representable floating point number during code verification. Consider the floating point value of 1.0. Previously, Polyspace represented this value as a range $0.9999 - 1.0001$. Now, Polyspace uses the exact value, that is 1.0.

Changes to verification results

None.

Changes to analysis options

- “New options” on page 8
- “Changes to existing options” on page 8

New options

None.

Changes to existing options

None.

Options removed

The `-machine-architecture` option has been removed. See “Verification uses native binary file” on page 5.

Polyspace Server for Ada Product

Improved Polyspace Metrics security with HTTPS

You can now configure the Polyspace Metrics Web server with a secure HTTPS protocol. This configuration enables encrypted communication between the Polyspace server and the Polyspace Metrics Web interface. See “Configure Polyspace Metrics Web Interface”.

R2012b

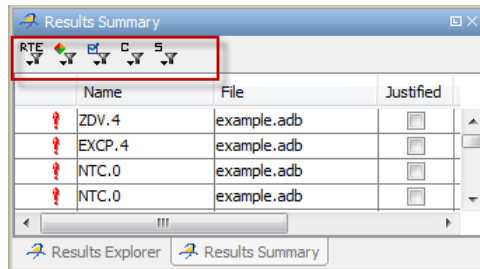
Version: 6.4
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Review of verification results improvement

Check filters in results summary view

The **Results Summary** toolbar provides check filters, which previously were available only in the **Results Explorer** view.



You can apply the check filters from either view. For example, if you filter checks by color and category in the **Results Summary** view, the software also filters out these checks from the **Results Explorer** view. For more information, see [Filtering Checks](#).

Justify and comment a group of checks

You can now select a group of checks in either the **Results Explorer** or **Result Summary** view, and then justify and add review information to those checks. For more information, see [Reviewing and Commenting Checks](#).

Variable values in tooltips

The display of variable values in **Source** view tooltips is improved, providing information that narrows the range of the variable. For example, the tooltip might indicate whether the variable values are multiples of a number. The following table has examples of how tooltips display variable range values.

Previously	R2012b
0 or 2 or 4 or [6 .. 2147483642 (0x7FFFFFFFA)]	even values in [0 .. 2147483642 (0x7FFFFFFFA)]
[-56 ..110] or [112 .. 166]	even values in [-56 .. 166]

Previously	R2012b
[-1265 .. -46] or -23 or 0	multiples of 23 in [-1265 .. 0]
[0 .. 22] or 44	0 or 22 or 44

You might also see the following type of tooltip for variables:

1008 or 4800 or 14400 or 23040 (values are multiples of 48)

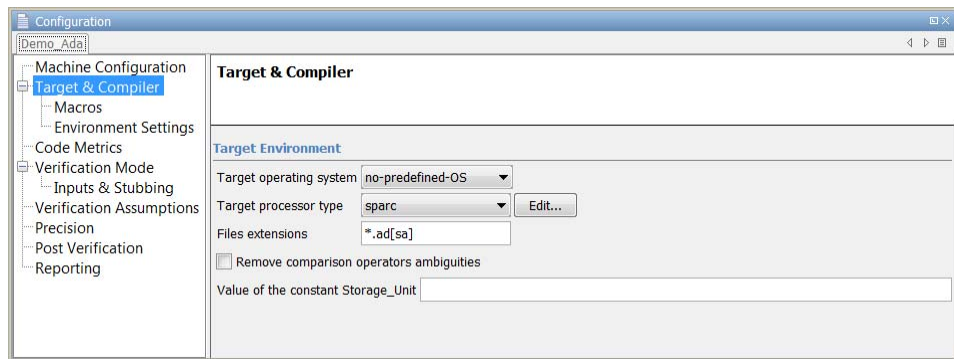
This enhancement might affect the contents of the text file *Your_Project_Variable_View.txt* file, which your verification generates in the `\results\Polyspace-Doc` folder. Instead of only an interval list for a variable, you might see, for example, the following conventions in the file:

- multiples of (a constant) in (the interval list)
- even values in (the interval list)
- odd values in (the interval list)

Reorganized Configuration pane

The Project Manager perspective of the Polyspace verification environment provides a reorganized **Configuration** pane that allows improved management of verification options.

The configuration process for code verification consists of different parts, for example, specifying your target environment and compiler behavior. The **Configuration** pane contains a tree with nodes that represent different parts of the configuration process. For example, to choose your target environment and compiler, select the **Target & Compiler** node and then specify your options.



You can specify the following option from the command line or through the **Configuration > Machine Configuration > Non-official options** field.

From command line	Removed from Configuration pane
-keep-all-files	Keep all preliminary results files

Polyspace plug-in for Eclipse with GNATbench

The Polyspace plug-in for the Eclipse IDE now supports Ada projects generated by GNATbench.

You can run verifications for Ada projects from the Eclipse IDE. However, no Polyspace settings are extracted from the Eclipse project.

Report content filtering

Previously, you used the MATLAB® Report Generator™ software to apply filters to report templates through only the **Run-time Check Details Ordered by Color/File** component. The software provides a new component, **Report Customization (Filtering)**, which allows you to apply filters from any point of the report hierarchy.

For more information, see Customizing Verification Reports.

Parent folder for verification results

You can now specify a parent folder for your verification results through the **Project and result folder** tab of the Polyspace Preferences dialog box. If you do not specify a parent folder, the software uses the active module folder as the parent folder. For more information, see [Specify Parent Folder for Results](#).

Support for relative paths

The Polyspace Project Manager now supports relative paths outside the project hierarchy. For example, paths for source files in a folder above the project location. Previously, relative paths were supported only for subfolders of the project.

Absolute paths are still supported.

Intermediate verification level support

From the Polyspace verification environment, you can no longer specify the value `CDFA` or `Control and Data Flow Analysis` for the option `-to`. If this value is specified in a project configuration file (`.cfg`), the software replaces the value with `compile` or `Source Compliance Checking`. However, from the command line, you can still specify `CDFA` or `Control and Data Flow Analysis` for `-to`.

Automatic import of comments and justifications

You can specify the batch option `-import-comments` *polyspace_results_folder_path* to automatically import comments and justifications from a previous verification. For more information, see [Batch Options](#).

Storage of temporary files

If you specify the new option `-tmp-dir-in-results-dir`, Polyspace does not use the standard `/tmp` or `C:\Temp` folder to store temporary files. Instead, Polyspace uses a subfolder of the results folder. If the results folder is mounted on a network drive, this action might affect processing speed. Use this option only when the temporary folder partition is not large enough and troubleshooting is required.

For more information, see [Batch Options](#).

Removal of Polyspace in One Click

The Polyspace in One Click feature will be removed in a future release.

Changes to verification results

None

Changes to analysis options

New options

None

Changes to existing options

Analysis options have been reorganized in the Project Manager. See “Reorganized Configuration pane” on page 17.

Option	Project Manager perspective			See also ...
	Previous label	R2012b label	Location on R2012b Configuration pane	
-add-to-results-repository	Add automatically to results repository	Add to results repository	Machine Configuration	
-array-expansion-size	Max size of global array variables	Unchanged	Precision	
-base-type-directly-visible	Remove comparison operators ambiguities	Unchanged	Target & Compiler	
-code-metrics	Calculate code metrics	Calculate code complexity metrics	Code Metrics	
-continue-with-all-niv	Continue after non initialized variables	Unchanged	Verification Assumptions	

Option	Project Manager perspective			See also ...
	Previous label	R2012b label	Location on R2012b Configuration pane	
-continue-with-in-out-niv	Continue with non-initialized in/out parameters	Unchanged	Verification Assumptions	
-critical-section-begin and -critical-section-end	Critical section details	Unchanged	Verification Mode	
-D	Defined Preprocessor Macros	Preprocessor definitions	Targets & Compler > Macros	
-data-range-specifications	Variable/function range setup	Unchanged	Verification Mode > Inputs & Stubbing	
-entry-points	Entry points or interruption	Entry points	Verification Mode	
-export-are-not-volatile	Treat export as non volatile	Unchanged	Verification Assumptions	
-extensions-for-spec-files	Files extensions	Unchanged	Target & Compiler	
-extra-flags		Non-official options	Machine Configuration	
-ignore-float-rounding	Ignore float rounding	Unchanged	Verification Assumptions	
-import-are-not-volatile	Treat import as non volatile	Unchanged	Verification Assumptions	

Option	Project Manager perspective			See also ...
	Previous label	R2012b label	Location on R2012b Configuration pane	
-init-stubbing-vars-random or -init-stubbing-vars-zero-or-random"	Initialize of uninitialized global variables	Initialization of uninitialized global variables	Verification Mode > Inputs & Stubbing	
-keep-all-files	Keep all preliminary results files	Command line only		
-known-NTC	Functions known to cause NTC	Procedures known to cause NTC	Verification Assumptions	
-machine-architecture	Run verification in 32 or 64-bit mode	Unchanged	Machine Configuration	
-main-generator	Generate a main	Verify module	Verification Mode	
-max-processes	Number of processes for multiple core system	Unchanged	Machine Configuration	
-modules-precision	Specific precision	Unchanged	Precision	
-no-automatic-stubbing	No automatic stubbing	Unchanged	Verification Mode > Inputs & Stubbing	
-O	Precision Level	Precision level	Precision	

Option	Project Manager perspective			See also ...
	Previous label	R2012b label	Location on R2012b Configuration pane	
-OS-target	Target operating system	Unchanged	Target & Compiler	
-path-sensitivity-delta	Improve Precision of interprocedural analysis	Unchanged	Precision	
-permissive	Permissive	Command line only		
-post-analysis-command	Command to apply after the end of the verification	Unchanged	Post Verification	
-pre-analysis-command	Command/script to apply before start of the code verification	Unchanged	Environment Settings	
-report-output-format	Report output format	Output format	Reporting	
-report-template	Report template	Report template name	Reporting	
-server	Send to Polyspace Server	Unchanged	Machine Configuration	
-storage-unit	Value of the constant Storage_Unit	Unchanged	Target & Compiler	

Option	Project Manager perspective			See also ...
	Previous label	R2012b label	Location on R2012b Configuration pane	
-strict	Strict	Command line only		
-target	Target processor type	Unchanged	Target & Compiler	
-temporal-exclusions-file	Temporal exclusion point	Temporally exclusive tasks	Verification Mode	
-timeout	Verification time limit	Unchanged	Precision	
-to	To end of	Verification level	Precision	“Intermediate verification level support” on page 22
-U	Undefined Preprocessor Macros	Undefine preprocessor definitions	Targets & Compiler > Macros	
-unit-by-unit	Run a verification unit by unit	Run unit by unit verification	Verification Mode	
-unit-by-unit-common-source	Unit by unit common source files	Unchanged	Verification Mode	
-variables-expansion-depth	Expansion limit for a structure	Unchanged	Precision	
-variables-to-expand	Variables to expand	Unchanged	Precision	

Options removed

None

Polyspace Server for Ada Product

Password-protected access to projects in Polyspace Metrics

You can now restrict access to a project by specifying a password:

- When you run a verification with Polyspace Metrics enabled or upload results to the Polyspace Metrics repository.
- After the creation of a project.

For more information, see [Protect Access to Project Metrics](#).

R2012a

Version: 6.3
New Features: Yes
Bug Fixes: Yes

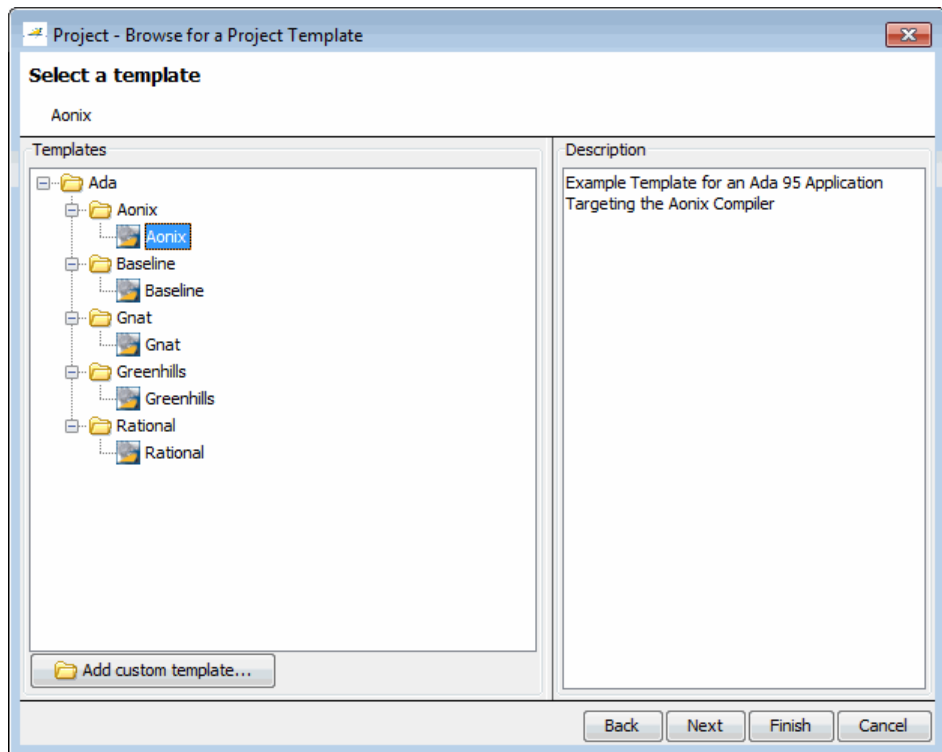
Polyspace Client for Ada Product

Compilation Environment Templates

Polyspace software now provides predefined compilation environment templates to help you configure verification projects.

These templates automatically set analysis options for the selected compiler, and help you locate the required include folders.

When creating a new project, you can select a template for your compiler.



For more information, see [Creating a Project](#).

Predefined Templates

Predefined templates are available for:

- **Aonix** – Ada 95 Application Targeting the Aonix Compiler
- **Baseline** – Generic Ada95 application
- **Gnat** – Ada 95 Application Targeting the Gnat Compiler
- **Greenhills** – Ada 95 Application Targeting the Greenhills Compiler
- **Rational** – Ada 95 Application Targeting the Rational Compiler

Custom Templates

You can also create custom templates from existing Project configurations, and use them to configure future projects.

For more information, see [Creating Custom Project Templates](#).

Suppression of NTC, NTL and UNR Checks Caused by Red Checks

Compatibility Considerations: Yes

Previously, the software would generate red NTC and NTL checks that were a consequence of other red checks. Now, the software does not generate these red checks. However, the software still gives the information that these red checks provided. The software highlights the corresponding call or loop identifier by applying a dashed underline to the identifier.

If the cause of the problem is known, the software provides this information in a tooltip for the underlined call or loop identifier. In addition, when you right-click the identifier, the context menu provides a **Go to Cause** item. Selecting this item takes you to the red check that is the cause.

The software:

- Does not generate gray UNR checks if the cause is a red check
- Still generates red NTC, NTL, and K-NTC checks for a call or loop identifier if the corresponding code contains orange checks.
- Does not generate NTC checks for the statements `exit` and `abort`, but provides tooltips for these statements. For example, `abort`, which does not correspond to an error, prevents the execution of a sequence of statements.

Compatibility Considerations

As a result of this new feature, you might observe a significant reduction in the total number of red and gray checks when compared to previous versions of the software.

Redefinition of Successful Verification

Previously, if a Polyspace verification failed, for example, during pass1 (Software Safety Analysis level 1), the software communicated the failure through log messages and the **Project Browser**. However, if you clicked the `xx_LAST_RESULTS.exe` file within the **Project Browser**, the software displayed any results (colored checks) that had been generated by the verification. Now, the software deems a verification successful provided some results have been generated.

Polyspace Report Generator Enhancements

You can:

- Generate multiple reports in the Results Manager perspective. See [Generating Verification Reports](#).
 - Customize report templates with MATLAB Report Generator software, which allows you to filter results by:
 - Justification status — Display all, justified, or unjustified checks.
 - Type — Display only listed types of run-time checks.
 - Function — Display only run-time checks from specified functions.
- For more information, see [Customizing Verification Reports](#).

Polyspace In One Click (POC) Enhancement

The POC software has been rewritten. The software that replaces the previous **Send To** functionality now runs verifications without requesting additional settings. See Using Polyspace In One Click.

Note Support for the **Send To** feature will be removed in a future release.

Header Files Without Run-Time Checks

It is quite common for code to contain header files with library inline functions that are never called. Previously, these files were listed in the **Results Explorer** view, which could slow down your review of results. Now, if header files do not contain run-time checks, the software does not list these header files in the **Results Explorer** view.

Improved Access to Polyspace Demos

In the Polyspace verification environment, you can now open supplied Demo projects through the **Help** menu.

Changes to Verification Results

Compatibility Considerations: Yes

Compatibility Considerations

Verification results might change when compared to previous versions of the software. Some checks might change color, and the Selectivity rate of your results might change.

NTC and NTL Checks

See “Suppression of NTC, NTL and UNR Checks Caused by Red Checks” on page 39.

Changes to Analysis Options

New Options

None

Changes to Existing Options

No name changes to existing options

Options Removed

None

Polyspace Server for Ada Product

Enhanced Polyspace Metrics Project Index

The enhanced project index enables you to display projects as categories, which is useful when you have a large number of projects to manage. Now, you can:

- Create multiple-level project categories.
- Move projects between categories by dragging and dropping projects.
- Rename and remove categories. You can remove categories without deleting the projects within the categories. The software moves these projects back to the root level.

For more information, see [Organizing Polyspace Metrics Projects](#).

Redefinition of Successful Verification

Previously, if a Polyspace verification failed, for example, during pass1 (Software Safety Analysis level 1), the software communicated the failure through log messages and the **Project Browser**. However, if you clicked the `xx_LAST_RESULTS.exe` file within the **Project Browser**, the software displayed any results (colored checks) that had been generated by the verification. Now, the software deems a verification successful provided some results have been generated.

R2011b

Version: 6.2
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Review Orange Checks that are Potential Run-Time Errors

Previously, there were two modes in which you could review verification results — manual and assistant. For the manual mode, you set the Assistant slider to **Off** and the software displayed all orange checks (in addition to the red and green checks) . With the assistant mode, there were three levels of review — corresponding to settings 1, 2, and 3 of the Assistant slider. You could specify the number of orange checks to display through the **Assistant Configuration** tab in the Polyspace Preference dialog box.

Now, Polyspace allows you to review results at five different levels. You can set the Review slider to 0, 1, 2, 3, or All:

- 0 — Display red and gray checks. In addition, display orange checks that are potential run-time errors. On the **Polyspace Preference > Review Configuration** tab, you can specify the type of potential run-time errors that you are interested in. You have the option of not displaying any orange checks.
- 1, 2, and 3 — This functionality is unchanged. Display red, gray, and green checks. In addition, display orange checks according to values specified on the **Polyspace Preference > Review Configuration** tab.
- All — Display red, gray, green, and all orange checks.

The **Assistant Configuration** tab is renamed the **Review Configuration** tab.

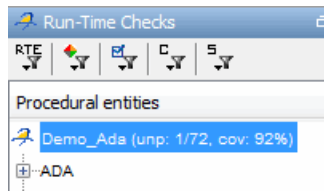
No Gray Checks in Unreachable Code

Compatibility Considerations: Yes

The only gray checks that Polyspace generates now are UNR checks for unreachable branches of code. In addition, Polyspace generates the UNR check only at the highest level of a branch. You no longer see nested UNR checks, that is, UNR checks in sub-branches.

In addition, the software displays two new metrics for the project in the procedural entities view:

- `unp` — Number of unreachable procedures (functions) as a fraction of the total number of procedures (functions)
- `cov` — Percentage of elementary operations in executable procedures (functions) covered by verification



These metrics provide:

- A measure of the code coverage achieved by the Polyspace verification.
- Indicators about the validity of your Polyspace configuration. For example, a large `unp` value and a low `cov` value may indicate an early red check or missing function call.

See Results Explorer Tab.

Compatibility Considerations

Due to the removal of non-UNR gray checks and nested UNR checks, verification results may change when compared to previous versions of the software.


Global Variable Range Information

In the **Variable Access** pane, Polyspace displays range information for read and write access operations on global variables within each source file. Previously, the displayed value was the union of all access operations on the global variable within a file. The software did not display range information for individual operations. Now, for global variables that are signed or unsigned integers, Polyspace also provides range information for the individual access operations from which the union value is obtained.

Variables	Detailed Type	Values
PKUTIL.INJECTION	$-2^{31}..2^{31}-1$	full-range $[-2^{31}..2^{31}-1]$
PKUTIL.COMPUTE_INJECTION		$[-2^{31}+1..2^{31}-1]$
PKUTIL.COMPUTE_INJECTION		full-range $[-2^{31}..2^{31}-1]$
PKUTIL.NEW_ALTITUDE	$-2^{31}..2^{31}-1$	12
PKUTIL.PARTIAL_INIT		12
PKTASKING.COMPUTE_NEW_COORDONATE		12
PKTASKING.COMPUTE_NEW_COORDONATE		12
PKUTIL.PARTIAL_INIT		12

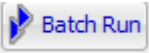
See Variable Access Pane.

Read and Write Access in Dead Code

If a read or write access operation on a global variable lies within dead code, then Polyspace colors the operation gray in the **Variable Access** pane. When you examine verification results, you can hide these operations by clicking the new filter button . See Variable Access Pane.

Run All Verifications in Project

You can have many verifications within a project, each verification being associated with an active configuration. Previously, you could only run one verification at a time from the Polyspace verification environment (PVE).

Now, if you select a project and click the button , Polyspace will run all verifications in the project. See [Running a Verification](#).

Green NIV check for Unchecked_Conversion Function

Previously, the software produced an orange NIV check for each call to an instance of the Ada generic library function `Unchecked_Conversion`. Now by default, the software produces a green NIV check for each call. If you want to revert to the previous behavior, run your verification with the option `-D POLYSPACE_UNCHECKED_CONVERSION_NO_INIT`.

Polyspace Server for Ada Product

Running Multiple Verifications Simultaneously

Compatibility Considerations: Yes

If you purchase more than one license for a Polyspace server, you can now configure the server to run multiple verifications at the same time. This can improve the performance of server verifications.

To configure your server to run multiple verifications, open the Remote Launcher Manager, then set the **Number of Polyspace verifications that can run simultaneously on this server** to the number of licenses you have activated for your server.

For more information, see Configuring Polyspace Server Software in the Polyspace Installation Guide.

Compatibility Considerations

If you configure your server to run more than one verification simultaneously, the server will not be able to run verifications using older versions of the software.

For example, if your server has both R2011a and R2011b software installed, you cannot run a verification using the R2011a software.

Polyspace Metrics

Review Changes between Results of Successive Verifications

You can specify a version of a project and review only the differences between verification results of the specified version and the previous verification. See [Review New Findings](#).

File Modules with Quality Levels

If you have projects with two or more file modules in the Polyspace verification environment, by default Polyspace Metrics displays verification results using the same module structure. However, Polyspace Metrics also allows you to create or delete file modules. You can group files into a module and specify a quality level for the module, which applies to all files within the module. This feature allows you to specify different quality levels for your files in the review of verification results. See [Creating a File Module and Specifying Quality Level](#).

Enhanced Graphs and Charts

Polyspace Metrics displays enhanced graphs and charts.

If you specify a range of project versions:

- On the **Summary** tab, **Run-Time Defects** are plotted as separate categories, High, Medium, and Low.
- On the **Run-Time Checks** tab:
 - Under **Confirmed Defects**, you see separate plots for the defect categories, High, Medium, and Low.
 - Under **Run-Time Findings**, you see separate plots for red checks, NTC checks, and gray checks.

If you specify a single version of a project, Polyspace Metrics displays file defect information, ordering the files according to the number of defects in each file. Use the new **# items** field to specify the maximum number of files for which information is displayed. See [Displaying Metrics for Single Project Version](#).

R2011a

Version: 6.1
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Code Metrics

Polyspace Metrics now generates Ada code metrics, giving you the number of:

- Protected shared variables
- Unprotected shared variables
- Files
- Lines of code
- Packages
- Packages that appear in with statements
- Subprograms that appear in with statements


You can view the metrics by:

- Using a Web browser to access the **Code Metrics** view of your project in Polyspace Metrics
- Examining the XML file that the software generates

For more information, see [Setting Up Verification to Generate Metrics and Review Code Metrics](#) in the Polyspace Products for Ada User's Guide.

Saving Polyspace Metrics Review

Previously, when you saved your project (**Ctrl+S**) after a review of results from Polyspace Metrics, the software would save your comments and justifications both locally and in the Polyspace Metrics repository.

Now, if you save your project (**Ctrl+S**), the software saves your review to a local folder only. A new button  is available on the Run-Time Checks toolbar. If you click this button, the software saves your comments and justifications to a local folder *and* the Polyspace Metrics repository.

This feature allows you to upload your review to the repository only when you are satisfied that your review is, for example, correct and complete.

You can still configure your software to display the previous behavior.

For more information, see Saving Review Comments and Justifications in the Polyspace Products for Ada User's Guide.

Support for Rational and Aonix Compilers

The software now provides support for the IBM Rational Apex and Aonix® compilers. For more information, see Operating system target for Standard Libraries compatibility in the Polyspace Products for Ada Reference.

Multi-Core Support

On multi-core computers, you can reduce verification time by specifying the use of core processors simultaneously to perform the verification. The software provides a new command line option `max-processes`, which uses a default value of 4. You can specify any value between 1 and 128. For more information, see [Number of processes for multiple CPU core systems](#) in the [Polyspace Products for Ada Reference](#).

Generated Main with Explicit Tasks and Accept Statements

Compatibility Considerations: Yes

If you select the option `Generate a main` and there are explicit tasks in the source code, then task bodies are verified like subprograms and accept statements are not executed. After verification, code associated with the accept statements is gray. For more information, see `Generate a main` in the Polyspace Products for Ada Reference.

Compatibility Considerations

Previously, if there were explicit tasks and accept statements in the source code, a verification run could have generated red, green, or orange checks for code within the accept statements. Now, verification colors the code within these accept statements gray.

Enhancements in Run-Time Checks Perspective

Compatibility Considerations: Yes

Within the source code view, you can investigate checks with tooltips that provide range information about variables.

In addition, when you click a check, the software provides information about the check in the Review Details pane.

For more information, see [Using Range Information in Results Manager Perspective](#) and [Selecting a Check to Review in the Polyspace Products for Ada User's Guide](#).

Compatibility Considerations

Because of these enhancements, the IPT and VOA checks are no longer required, and the software does not generate these checks anymore.

UOVFL and UNFL Checks Removed

Compatibility Considerations: Yes

The software no longer generates UOVFL and UNFL checks, but generates OVFL checks in place of these checks.

Compatibility Considerations

Due to the replacement of UOVFL and UNFL checks by the OVFL check, the number of green checks may decrease when compared with previous versions of the software. For example, a combination of an orange OVFL and a green UNFL generated by a previous version may be replaced by a single orange OVFL.

NIV Checks for Universal Constants

Compatibility Considerations: Yes

The software now generates NIV checks for read operations on universal constants. If the constants are used in your code, the NIV checks are green. If the constants are in unreachable code, the NIV checks are gray.

Compatibility Considerations

This enhancement may change the check metrics and selectivity of the verification. The number of green and gray checks may be higher compared to the number generated by previous versions of the software.

Variable Range Inconsistency between Variable Access Pane and Tooltips

The range given for a variable in the Variable Access Pane (Variables View) can differ from the range given by tooltips on the reads of a variable in the Source code view. The range provided by the tooltip will be wider than the range given in the Variables View.

This difference is due to imprecision in the tooltip. The Variables View provides the correct range for the variable.

For example:

- Variables View states that variable X is in range $[0..4000]$
- Tooltip on a read of X states that the range is $[0,7000]$.

In this case, $[0..4000]$ is the correct range. The tooltip range is caused by imprecision that may be fixed in future releases.

Verification Time Limit

You can now specify a time limit for verifications using the `-timeout` option. If the verification does not complete within the specified time, the verification fails.

For more information, see “Verification Time Limit” in the Polyspace Products for Ada Reference.

Automatic Addition of Specifications for Selected Source Files

When launching a verification from the Eclipse IDE or the Polyspace Verification Environment, the software automatically searches for the package specifications associated with the selected source files, and adds them to the set of sources to verify.

As a result, the verification may contain more source files than you select.

Stubbed Tasks

Programs with stubbed tasks, such as those using Ada rendezvous, previously caused compilation errors. These programs can now be verified.

Scaling Issue for Large Applications with Nested Structures/Arrays

Compatibility Considerations: Yes

With R2011a, you may experience scaling problems for large applications that manipulate strongly nested structures or arrays. When verifying such applications, the verification may fail during the “Software Safety Analysis Level 0” phase. No verification results are generated, although the data dictionaries (Variable View and Call-Graph View) are accessible.

With previous releases, such applications could be verified, but the verification required several days to complete, and produced results with very low selectivity.

If you experience this problem, MathWorks recommends performing a unit-by-unit verification. For more information, see *Running Verification Unit-by-Unit* in the *Polyspace Products for Ada User’s Guide*.

Compatibility Considerations

Verification may fail for code that was previously verified with an earlier version of the product.

Product Name Change in Files and Folders

Compatibility Considerations: Yes

The Polyspace product name has changed from “PolySpace” to “Polyspace” in R2011a. This change impacts the name of all files and folders created by the software.

For example:

- PolySpace-Doc folder has changed to Polyspace-Doc
- PolySpace_xxxx.log file has changed to Polyspace_xxxx.log

Compatibility Considerations

If you have existing folders that use the previous product name (for example, PolySpace/PolySpace_Common) the R2011a installation will continue to use these existing folders. However, any files or folders created during or after installation will use the new name.

If you have any shortcuts or scripts that are case-sensitive, you should update them to use the correct name.

License Manager Support

The License Manager for Polyspace products has been upgraded to FlexNet® 11.9.

You may need to upgrade your FlexNet server and daemon.

For more information, see Polyspace License Installation in the Polyspace Installation Guide.

Changes to Verification Results

Compatibility Considerations: Yes

- “Write Access in Data Table with Main Generator and Protected Objects” on page 80
- “NIV for Variables Initialized at Declaration” on page 81
- “Range Error with greenhills OS Target” on page 81
- “UOVFL and UNFL Checks Removed” on page 81
- “NIV Checks for Universal Constants” on page 81
- “Constants Defined in Package System” on page 82
- “Initialization of Variables Declared and Assigned in Specs” on page 82
- “Parameterless Protected Procedure as Entry Point” on page 82

Compatibility Considerations

Verification results may change when compared to previous versions of the software. Some checks may change color, and the Selectivity rate of your results may change.

Refer to the following sections for information on the specific changes.

Write Access in Data Table with Main Generator and Protected Objects

In previous releases, when using the `-main-generator` option, an incorrect write access could appear in the data-table when entries of protected objects were called.

In R2011a, the data-table no longer includes locations with critical section names.

There may be less write accesses with protected objects that have defined entries when you use the option `-main-generator`.

NIV for Variables Initialized at Declaration

In previous releases, there is no NIV check when a variable is initialized at the declaration.

In R2011a, verification always puts NIV checks on variables and constants. OVFL checks are not put on constants.

The total number of checks may change when compared with previous releases.

Range Error with greenhills OS Target

In previous releases, verification could report a false red when using “Ada.interrupts.Interrupt_ID” in source code designed for the greenhills compiler.

In R2011a, when you use the type `Ada.interrupts.Interrupt_ID`, and set the `-OS-target` to `greenhills`, the bounds of the type `Interrupt_ID` have changed.

Verification results may change when compared to previous versions of the software. Some checks may change color, and the Selectivity rate of your results may change.

UOVFL and UNFL Checks Removed

The software no longer generates UOVFL and UNFL checks, but generates OVFL checks in place of these checks.

The number of checks may decrease compared to previous versions of the software.

NIV Checks for Universal Constants

The software now generates NIV checks for read operations on universal constants. If the constants are used in your code, the NIV checks are green. If the constants are in unreachable code, the NIV checks are gray.

Verification results may change when compared to previous versions of the software. The number of green and gray checks may be higher compared to the number generated by previous versions of the software.

Constants Defined in Package System

In previous releases, constants defined in package system were ignored.

For example:

```
Max_Binary_Modulus      : constant := 16#100000000#;  
Max_Nonbinary_Modulus  : constant := 16#FFFFFFFF#;
```

In R2011a, values of constants defined in the package system that are used in the program to be verified are taken into account for the verification. This may impact the results.

Initialization of Variables Declared and Assigned in Specs

In previous releases, verification could incorrectly report a red ZDV check for a variable of a type with default values, when the variable is global.

In R2011a, when using the option `-main-generator`, components with initial values of global variables of composite types that are not initialized are considered full range.

Verification results may change when compared to previous versions of the software. Some checks may change color, and the Selectivity rate of your results may change.

Parameterless Protected Procedure as Entry Point

In previous releases, when a parameterless protected procedure is used as an entry point via the option `-entry-points`, the verification fails.

In R2011a, parameterless protected procedures applied to a global object are now accepted as values of the option `-entry-points`.

Changes to Analysis Options

New Options

- **Calculate code metrics** (-code metrics) – See “Code Metrics” on page 65.
- **Verification time limit** (-timeout) – See “Verification Time Limit” on page 74.
- **Number of processes for multiple CPU core systems** (-max-processes) – See “Multi-Core Support” on page 68.
- **Less range information** (-less-range-information) – See “Enhancements in Run-Time Checks Perspective” on page 70.

Changes to Existing Options

None.

Deprecated Options

None.

Polyspace Server for Ada Product

Code Metrics

Polyspace Metrics now generates Ada code metrics, giving you the number of:

- Protected shared variables
- Unprotected shared variables
- Files
- Lines of code
- Packages
- Packages that appear in with statements
- Subprograms that appear in with statements


You can view the metrics by:

- Using a Web browser to access the **Code Metrics** view of your project in Polyspace Metrics
- Examining the XML file that the software generates

For more information, see [Setting Up Verification to Generate Metrics and Review Code Metrics](#) in the Polyspace Products for Ada User's Guide.

Saving Polyspace Metrics Review

Previously, when you saved your project (**Ctrl+S**) after a review of results from Polyspace Metrics, the software would save your comments and justifications both locally and in the Polyspace Metrics repository.

Now, if you save your project (**Ctrl+S**), the software saves your review to a local folder only. A new button  is available on the Run-Time Checks toolbar. If you click this button, the software saves your comments and justifications to a local folder *and* the Polyspace Metrics repository.

This feature allows you to upload your review to the repository only when you are satisfied that your review is, for example, correct and complete.

You can still configure your software to display the previous behavior.

For more information, see Saving Review Comments and Justifications in the Polyspace Products for Ada User's Guide.

Multi-Core Support

On multi-core computers, you can reduce verification time by specifying the use of core processors simultaneously to perform the verification. The software provides a new command line option `--max-processes`, which uses a default value of 4. You can specify any value between 1 and 128.

For more information, see [Number of processes for multiple CPU core systems in the Polyspace Products for Ada Reference](#).

Generated Main with Explicit Tasks and Accept Statements

Compatibility Considerations: Yes

If you select the option `Generate a main` and there are explicit tasks in the source code, then task bodies are verified like subprograms and `accept` statements are not executed. After verification, code associated with the `accept` statements is gray. For more information, see `Generate a main` in the Polyspace Products for Ada Reference.

Compatibility Considerations

Previously, if there were explicit tasks and `accept` statements in the source code, a verification run could have generated red, green, or orange checks for code within the `accept` statements. Now, verification colors the code within these `accept` statements gray.

Automatic Comment Import for Server Verifications

When you download results from the Polyspace server, the software now automatically imports any comments from results in the destination folder into the downloaded results (except for verifications using the option `-add-to-results-repository`).

As a result of this change, you can now download intermediate results for a verification running on the Polyspace server, and add or edit comments on those results. When you later download the final results, your comments are preserved.

You can also download and comment on a single unit of a unit-by-unit verification, even if other units are still pending in the server queue. When you download the final results (which overwrites the earlier results), your comments are preserved.

License Manager Support

The License Manager for Polyspace products has been upgraded to FlexNet 11.9.

You may need to upgrade your FlexNet server and daemon.

For more information, see Polyspace License Installation in the Polyspace Installation Guide.

R2010b

Version: 6.0
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Polyspace Graphical User Interface

Redesigned Polyspace graphical user interface replaces the Launcher and Viewer modules with a single, unified interface called the Polyspace verification environment (PVE).

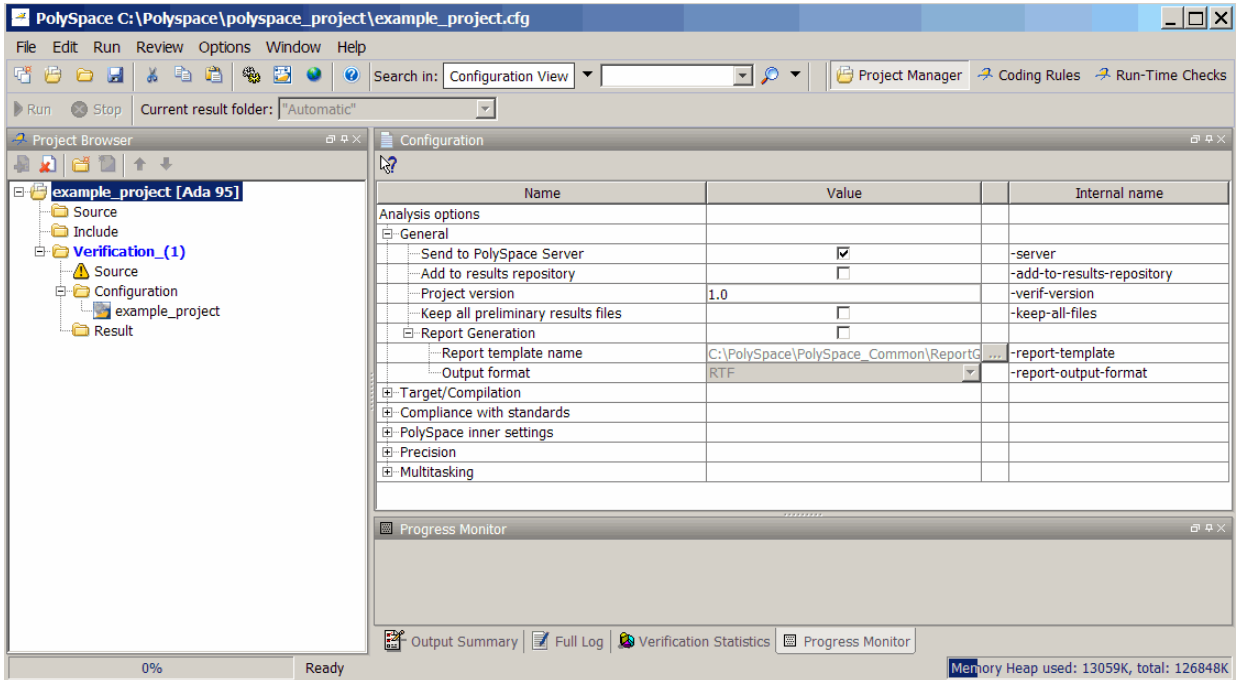
You use the Polyspace verification environment to create Polyspace projects, launch verifications, and review verification results. The new interface also enables you to provide comments in the source code or in the results.

The Polyspace verification environment consists of two perspectives:

- “Project Manager Perspective” on page 94
- “Run-Time Checks Perspective” on page 95

Project Manager Perspective

The Project Manager perspective allows you to create projects, set verification parameters, and launch verifications.



For information on using the Project Manager perspective, see *Setting Up a Verification Project in the Polyspace Products for Ada User's Guide*.

Run-Time Checks Perspective

The Run-Time Checks perspective allows you to review verification results, comment individual checks, and track review progress.

The screenshot displays the Polyspace Run-Time Checks perspective. The interface includes a menu bar (File, Edit, Run, Review, Options, Window, Help), a toolbar with search and project management tools, and a main workspace divided into several panels:

- Procedural entities:** A tree view on the left showing the project structure. A checkmark icon is visible next to the selected entity.
- Review Details:** A central panel showing the selected check. It includes the file path (example.adb / PROCEDURE_ZDV / line 34 / column 30), the source code snippet, and the error message: "Error : float division by zero occurs".
- Review Statistics:** A table on the right showing the progress of coding reviews.

Coding review progress	Count	Progr...
Red ZDV reviewed / to review	0/1	0
Red reviewed / to review	0/7	0
Gray reviewed / to review	0/3	0
Orange reviewed / to review	0/5	0
Software reliability indicator	39/60	65
- Source:** A window showing the full source code of the selected procedure.
- Call Hierarchy:** A window showing the call stack for the selected code.
- Variable Access:** A window showing the variables accessed in the selected code.

Annotations with arrows point to specific elements:

- Selected check:** Points to the error message in the Review Details panel.
- Coding review:** Points to the Review Statistics table.
- Procedural entities:** Points to the tree view on the left.
- Source code:** Points to the source code window.
- Variables:** Points to the Variable Access window.
- Call tree:** Points to the Call Hierarchy window.

For information on using the Run-Time Checks perspective, see *Reviewing Verification Results in the Polyspace Products for Ada User's Guide*.

Data Range Specifications

The Data Range Specifications (DRS) feature allows you to set constraints on data ranges using a text file. With this text file, you can specify data ranges for global variables, stubbed functions and procedures, and function call inputs.

For more information, see [Specifying Data Ranges for Variables, Functions, and Procedures](#) in the Polyspace Products for Ada User's Guide.

Extended Support for Tasks

You can now verify Ada code that contains the following task-related features :

- Pointers to explicit tasks
- Private task types
- Tasks with discriminants
- Entry families
- Explicit tasks declared in a package – you can now verify the package with the `-main-generator` option

Previously, these features would generate errors during a verification.

Preprocessor Macros for Compilation

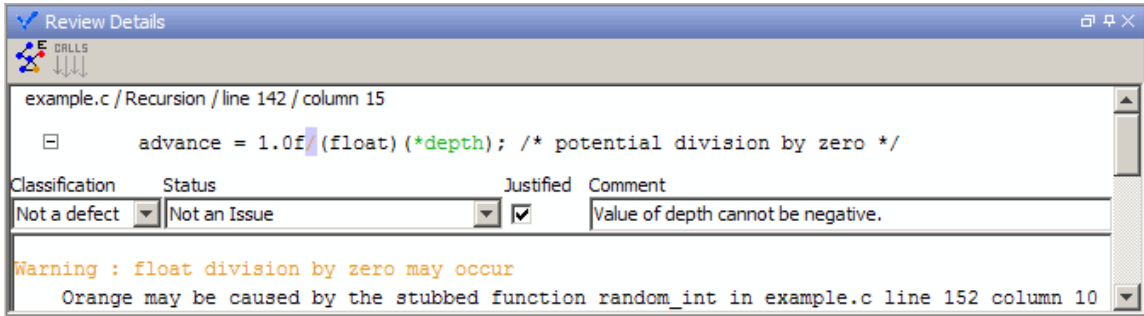
You can now specify, with the `-D` option, the use of preprocessor macros in code compilation. The `-U` option, which nullifies this `-D` option, is also available.

For more information, see *Defined Preprocessor Macros(-D compiler-flag)* and *Undefined Preprocessor Macros (-U compiler-flag)* in the *Polyspace Products for Ada Reference Guide*.

New Options to Classify Run-Time Checks

Compatibility Considerations: Yes

When reviewing results in the Run-Time Checks perspective, the software now provides additional options for classifying checks



After you review the check, you can specify the following:

- **Classification** – Select an option to describe the seriousness of the issue.
- **Status** – Select an option to describe how you intend to address the issue.
- **Justified** – Select the check box to indicate that you have justified this check.
- **Comment** – Enter additional information about the check

The software provides pre-defined values for Classification and Status. You can also define your own statuses.

In addition to reviewing checks through the user interface, you can place comments in your code that highlight and categorize checks identified in previous verifications. The software displays the information that you provide within your code comments, and marks the checks as **Justified**.

For more information, see *Reviewing and Commenting Checks* in the *Polyspace Products for Ada User's Guide*.

Compatibility Considerations

The syntax for code comments has changed to reflect the new options for categorizing checks.

The syntax for run-time checks is now:

```
-- polyspace<RTE:RTE1 : [Classification] : [Status] > [Comment]
```

If you placed comments in your code using the previous syntax, the comments will still appear in your results, but the text may be displayed in different columns.

For more information, see [Syntax for Run-Time Errors in the Polyspace Products for Ada User's Guide](#).

Permissiveness on File and Folder Names

Polyspace software now allows space characters in the names of Projects, source files, and folders, as well as in option arguments.

In addition, multiple source files with the same name are now allowed.

Note Non-ASCII characters in file names are not supported.

Default Target Processor

Compatibility Considerations: Yes

The default setting of the Target processor type (`-target-processor`) option has changed from SPARC to i386.

Compatibility Considerations

If you launch verifications without specifying a value for this option, the default value has changed. Therefore, your verification results may change when compared to previous versions of the software. Some checks may change color, and the Selectivity rate of your results may change.

Operating System Support

Added support for the Windows® 7 operating system.

Solaris™ operating system is no longer supported for new installations.

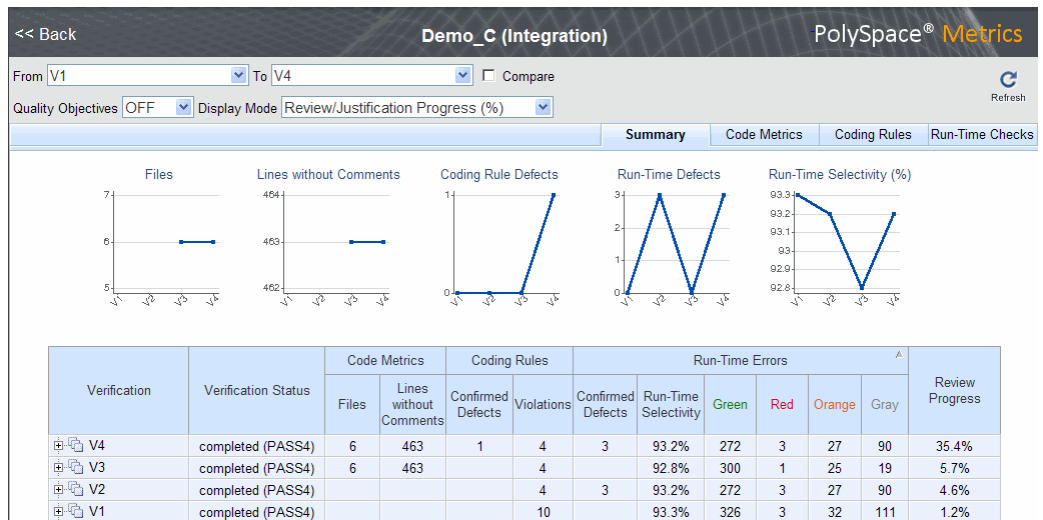
For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

Polyspace Metrics Web Interface

A web-based tool for software development managers, quality assurance engineers, and software developers, which allows you to do the following in software projects:

- Evaluate software quality metrics
- Monitor the variation of code metrics, and run-time checks through the lifecycle of a project
- View defect numbers, run-time reliability of the software, review progress, and the status of the code with respect to software quality objectives.



In addition, if you have the Polyspace Client™ for Ada product installed on your computer, you can drill down to run-time checks in the Polyspace verification environment. You can review these run-time checks and, if required, classify these checks as defects.

For more information, see Software Quality with Polyspace Metrics in the Polyspace Products for Ada User's Guide.

Automatic Verification

Configure verifications to start automatically and periodically, for example, at a specific time every night. At the end of each verification, the software stores results in a results repository and updates the metrics for your software project. You can also configure the software to send you an email at the end of the verification. This email contains links to results, compilation errors, run-time errors, or processing errors.

For more information, see [Specifying Automatic Verification in the Polyspace Products for Ada User's Guide](#).

Operating System Support

Added support for the Windows 7 operating system.

Solaris operating system is no longer supported for new installations.

For more information, see the Polyspace Installation Guide.

R2010a

Version: 5.5
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

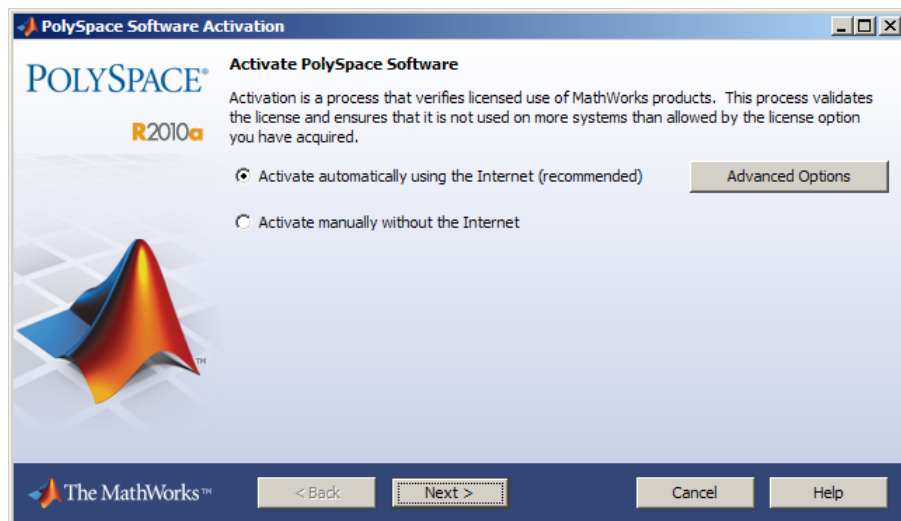
License Activation

Polyspace products now support the MathWorks software activation mechanism.

Activation is a process that verifies licensed use of MathWorks® products. The process validates your product licenses and ensures that they are used correctly. You must complete the activation process before you can use Polyspace software.

Note If you are using Designated Computer (Individual) licenses, you must activate the license for each Polyspace system individually. However, if you are using Concurrent licenses for multiple Polyspace systems, you do not need to activate each Polyspace system. You activate the license once (for the FLEXnet license server), then provide license files for each Polyspace system.

The easiest way to activate the software is to log in to your MathWorks Account during installation. At the end of the installation process, the Polyspace Software Activation dialog box opens.



Follow the prompts in the Polyspace Software Activation dialog box to complete the activation process.

If you do not have a MathWorks account, you can create one during the activation process. To create an account, you must have an Activation Key, which identifies the license you want to install and activate.

If your Polyspace system is not connected to the internet, you can access the MathWorks License Center on a computer with internet access, activate your license, and download a license file for transfer to your Polyspace system. If you do not have access to a computer with an Internet connection, contact Customer Support.

For more information on how to activate your software, see *Activating Polyspace Software* in the Polyspace Installation Guide.

For more information on software activation, including frequently asked questions, refer to the MathWorks Web site:
www.mathworks.com/support/activation/polyspace.html

Source Code Comment Support

Polyspace software now allows you to place comments in your code that provide information about known run-time errors. You can use these comments to highlight and categorize previously identified run-time errors. This information can then make the review process quicker and easier.

When you review verification results, the Viewer displays comments on individual checks. You can then skip these commented checks during the review process, or simply use them as additional information during your review.

For more information, see [Highlighting Known Run-Time Errors](#) in the Polyspace Products for Ada User's Guide.

Eclipse Integration

Polyspace integration with the Eclipse IDE, Version 3.4 and 3.5.

The Polyspace Client for Ada can be integrated with the Eclipse™ Integrated Development Environment through the Polyspace plug-in for Eclipse IDE.

This plug-in provides Polyspace source code verification and bug detection functionality for source code developed within Eclipse IDE. Features include the following:

- A contextual menu that allows you to launch a verification of one or more files.
- Views in the Eclipse editor that allow you to set verification parameters and monitor verification progress.

For more information, see *Using Polyspace Software in the Eclipse IDE* in the *Polyspace Products for Ada User's Guide*.

Operating System Support

Added support for the following Linux distributions:

- OpenSuSE 11.1
- Debian 5.x
- Ubuntu 8.04, 8.10, 9.04, and 9.10

For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

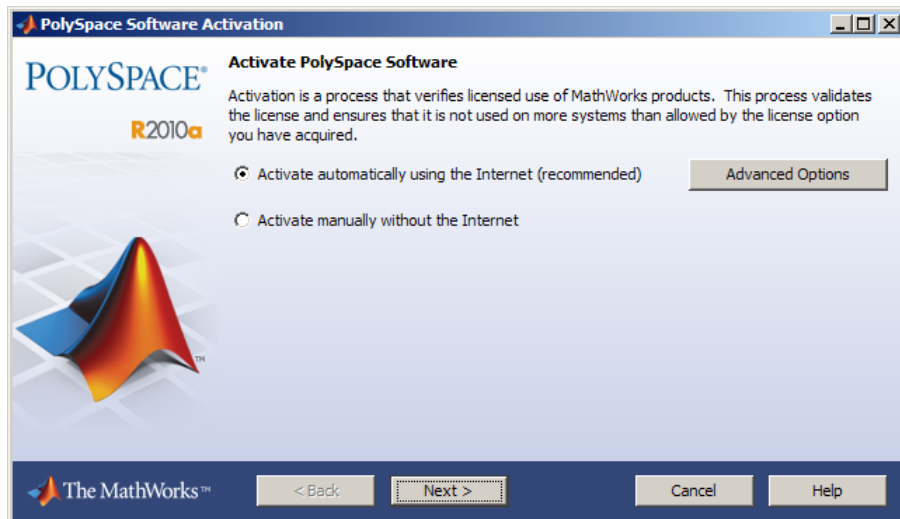
License Activation

Polyspace products now support the MathWorks software activation mechanism.

Activation is a process that verifies licensed use of MathWorks products. The process validates your product licenses and ensures that they are used correctly. You must complete the activation process before you can use Polyspace software.

Note If you are using Designated Computer (Individual) licenses, you must activate the license for each Polyspace system individually. However, if you are using Concurrent licenses for multiple Polyspace systems, you do not need to activate each Polyspace system. You activate the license once (for the FLEXnet license server), then provide license files for each Polyspace system.

The easiest way to activate the software is to log in to your MathWorks Account during installation. At the end of the installation process, the Polyspace Software Activation dialog box opens.



Follow the prompts in the Polyspace Software Activation dialog box to complete the activation process.

If you do not have a MathWorks account, you can create one during the activation process. To create an account, you must have an Activation Key, which identifies the license you want to install and activate.

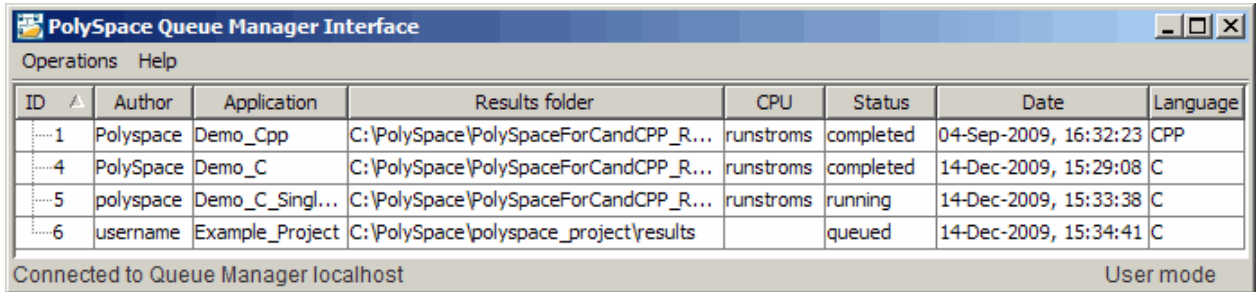
If your Polyspace system is not connected to the internet, you can access the MathWorks License Center on a computer with internet access, activate your license, and download a license file for transfer to your Polyspace system. If you do not have access to a computer with an Internet connection, contact Customer Support.

For more information on how to activate your software, see *Activating Polyspace Software* in the Polyspace Installation Guide.

For more information on software activation, including frequently asked questions, refer to the MathWorks Web site:
www.mathworks.com/support/activation/polyspace.html

Queue Manager Interface

The Polyspace Queue Manager Interface (Spooler) is now available on Linux machines, providing a graphical interface for managing verification jobs on the Polyspace server.



The screenshot shows a window titled "PolySpace Queue Manager Interface" with a menu bar containing "Operations" and "Help". Below the menu bar is a table with the following columns: ID, Author, Application, Results folder, CPU, Status, Date, and Language. The table contains four rows of data. At the bottom of the window, it displays "Connected to Queue Manager localhost" on the left and "User mode" on the right.

ID	Author	Application	Results folder	CPU	Status	Date	Language
1	Polyspace	Demo_Cpp	C:\PolySpace\PolySpaceForCandCPP_R...	runstroms	completed	04-Sep-2009, 16:32:23	CPP
4	PolySpace	Demo_C	C:\PolySpace\PolySpaceForCandCPP_R...	runstroms	completed	14-Dec-2009, 15:29:08	C
5	polyspace	Demo_C_Singl...	C:\PolySpace\PolySpaceForCandCPP_R...	runstroms	running	14-Dec-2009, 15:33:38	C
6	username	Example_Project	C:\PolySpace\polyspace_project\results		queued	14-Dec-2009, 15:34:41	C

Connected to Queue Manager localhost User mode

For more information, see [Managing Verification Jobs Using Polyspace Queue Manager](#) in the Polyspace Products for Ada User's Guide.

Operating System Support

Added support for the following Linux distributions:

- OpenSuSE 11.1
- Debian 5.x
- Ubuntu 8.04, 8.10, 9.04, and 9.10

For more information, see the Polyspace Installation Guide.

R2009b

Version: 5.4
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Report Generator

New Report Generator that presents Polyspace results in PDF, HTML, and other output formats.

The Polyspace Report Generator allows you to generate reports about your verification results, using the following predefined report templates:

- **Coding Rules Report** – Provides information about compliance with MISRA-C Coding Rules, as well as Polyspace configuration settings for the verification.
- **Developer Report** – Provides information useful to developers, including summary results, detailed lists of red, orange, and gray checks, and Polyspace configuration settings for the verification.
- **Developer with Green Checks Report** – Provides the same content as the Developer Report, but also includes a detailed list of green checks.
- **Quality Report** – Provides information useful to quality engineers, including summary results, statistics about the code, graphs showing distributions of checks per file, and Polyspace configuration settings for the verification.

The Polyspace Report Generator allows you to generate verification reports in the following formats:

- HTML
- PDF
- RTF
- Microsoft® Word
- XML

Note Microsoft Word format is not available on UNIX platforms. RTF format is used instead.

For more information, see [Generating Reports of Verification Results in the Polyspace Products for Ada User's Guide](#) .

Main Generator Enhancements

Compatibility Considerations: Yes

Enhanced main generator that considers the scope of a procedure and variable, improving error detection at the package level.

This change may affect your results compared with previous releases, and how you interpret the new results. Specific changes include:


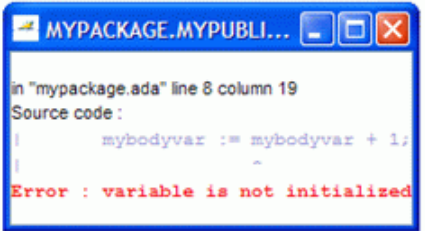
- Uninitialized package body variables are considered uninitialized
- Uncalled package-scope procedures/functions are considered unreachable
- Functions/procedures declared at the spec level are called only once
- Uninitialized spec level variables are considered possibly uninitialized

For more information on the main generator, see Main Generator Overview in the Polyspace Products for Ada User's Guide .

Uninitialized Package Body Variables are Considered Uninitialized

Uninitialized variables that are declared only in the package body are now considered uninitialized, and generate red NIV checks.

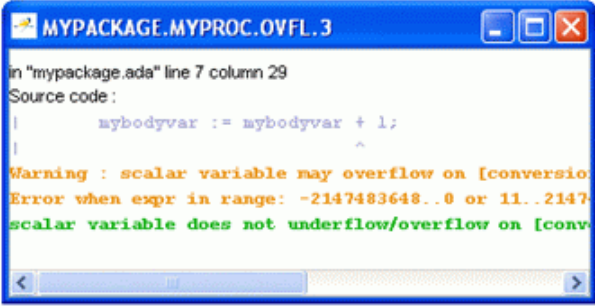
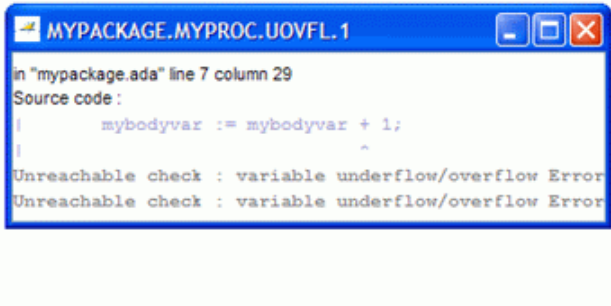
Previously, these variables were considered initialized (green NIV) with full-range values. This behaviour made interpretation of results easier and allowed verification to continue. However, the software now considers these variables uninitialized (red NIV) and stops verification at this point. This new behaviour is more accurate with respect to the actual initialization state of the variables. You must correct the code before verification can continue. Alternatively, you can use the option `-continue-with-all-niv`.

Change 1: Uninitialized package body variables are considered uninitialized	
Settings: -main-generator	
9a	9b
<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 procedure myPublicProc; 4 end mypackage; 5 package body mypackage is 6 mybodyvar : myint; 7 procedure myPublicProc is begin 8 mybodyvar := mybodyvar + 1; 9 end myPublicProc; 10 end mypackage; </pre>	<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 procedure myPublicProc; 4 end mypackage; 5 package body mypackage is 6 mybodyvar : myint; 7 procedure myPublicProc is begin 8 mybodyvar := mybodyvar + 1; 9 end myPublicProc; 10 end mypackage; </pre>
	

Uncalled Package-scope Procedures/Functions are Considered Unreachable

Procedures and functions that are declared in the package but not called by code within the package body provided for the verification will now be considered unreachable (gray).

Previously, all procedures and functions in a package were considered for verification and subsequently colored. The argument for this behavior was that these functions and procedures could be called by code inside the package that had not been provided for the verification. Now, the software considers this code unreachable (gray) unless there is a path of execution that leads to it.

Change 2: Uncalled package-scope procedures/functions are considered unreachable	
Settings: -main-generator	
9a	9b
<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 end mypackage; 4 package body mypackage is 5 mybodyvar : myint; 6 procedure myProc is begin 7 mybodyvar := mybodyvar + 1; 8 end myProc; 9 end mypackage; </pre> 	<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 end mypackage; 4 package body mypackage is 5 mybodyvar : myint; 6 procedure myProc is begin 7 mybodyvar := mybodyvar + 1; 8 end myProc; 9 end mypackage; </pre> 

Functions/Procedures Declared at the Spec Level are Called Only Once

Functions or procedures declared at the specification level are called only once.


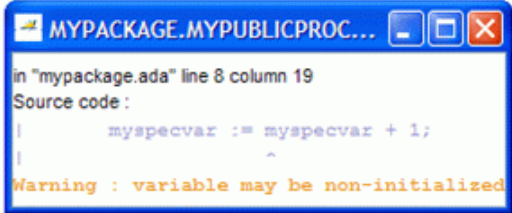
Note This behavior changed in Release 2010a. In R2010a or later, the main generator can call a function several times.

Uninitialized Spec Level Variables are Considered Possibly Uninitialized

Uninitialized variables declared in a package specification will now be considered possibly uninitialized (orange NIV). Previously, these variables were considered initialized (green NIV) with full-range values.

The software now considers uninitialized variables that are declared only in the package *body* as uninitialized (red NIV). However, for uninitialized variables declared in a package *specification*, it is possible that packages that use these variables may initialize these variables. The software now recognizes this possibility and generates orange NIV checks for uninitialized variables declared in a package specification.

This behavior is not changed if you use options `-init-stubbing-vars-random` or `-init-stubbing-vars-zero-or-random` to initialize uninitialized variables. Specification-level variables will still be considered possibly uninitialized (orange NIV), because the packages that use these variables can alter the variables, even to the extent of uninitialized the variables.

Change 4: Uninitialized spec level variables are considered possibly uninitialized	
Settings: -main-generator	
9a	9b
<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 procedure myPublicProc; 4 <u>myspecvar</u> : myint; 5 end mypackage; 6 package body mypackage is 7 procedure myPublicProc is begin 8 <u>myspecvar</u> := <u>myspecvar</u> + 1; 9 end myPublicProc; 10 end mypackage; </pre>	<pre> 1 package mypackage is 2 type myint is new integer range 1..10; 3 procedure myPublicProc; 4 <u>myspecvar</u> : myint; 5 end mypackage; 6 package body mypackage is 7 procedure myPublicProc is begin 8 <u>myspecvar</u> := <u>myspecvar</u> + 1; 9 end myPublicProc; 10 end mypackage; </pre>
 <p>in "mypackage.ada" line 8 column 19 Source code : myspecvar := myspecvar + 1; ^ variable is initialized</p>	 <p>in "mypackage.ada" line 8 column 19 Source code : myspecvar := myspecvar + 1; ^ Warning : variable may be non-initialized</p>

Compatibility Considerations

Changes to the main generator may result in differences in your verification results, when compared with earlier versions of the software. If you verified your code with previous versions of the software (for example, R2009a), be aware of these changes, how they affect your colored results and the way you interpret the results.

Global Data Graphs

New Graphs (similar to concurrent access graphs) available for all global data.

You can display the access sequence for any variable that is read or written in the code. The access graph displays the read and write access for the variable.

For more information, see [Displaying the Access Graph for Variables in the Polyspace Products for Ada User's Guide](#) .

Unit-by-unit Verification

New option to create a separate verification job for each source file in the project.

When you run a unit-by-unit verification, each source file is compiled, sent to the Polyspace Server, and verified individually.

The queue manager displays a job for the full verification group, as well as jobs for each unit (using a tree structure).

When verification is complete, you can download and view results for the entire project, or for individual units. When downloading a verification group, all the unit results are downloaded and a summary of the download status for each unit is displayed.

Note Unit by unit verification is available only for server verifications.

For more information, see [Running Verification Unit-by-Unit](#) in the Polyspace Products for Ada Reference.

Operating System Support

Added support for Windows Server® 2008.

For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

Operating System Support

Added support for Windows Server 2008.

For more information, see the Polyspace Installation Guide.

R2009a

Version: 5.3
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Character Encoding Options

New character encoding option allows you to view source files created on an operating system that uses different character encoding than your current system.

You specify the character encoding used by the operating system on which the source file was created using the **Character encoding** tab in the Preferences dialog box of the Polyspace Viewer.

For more information, see [Setting Character Encoding Preferences](#) in the Polyspace Products for Ada User's Guide.

Operating System Support

Added support for Windows Server 2003, Windows Vista™, and Red Hat Enterprise Linux Workstation v.5.

For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

Operating System Support

Added support for Windows Server 2003, Windows Vista, and Red Hat Enterprise Linux Workstation v.5.

For more information, see the Polyspace Installation Guide.

R2008b

Version: 5.2
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Operating System Support

Added support for 64-bit Linux.

For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

Operating System Support

Added support for 64-bit Linux.

For more information, see the Polyspace Installation Guide.

R2008a

Version: 5.1
New Features: Yes
Bug Fixes: Yes

Polyspace Client for Ada Product

Removed Cygwin Software Dependency for Windows Platforms

Compatibility Considerations: Yes

Previous versions of Polyspace products used Cygwin™ emulation to run UNIX® commands on Windows systems.

In version 5.1, the Cygwin software dependency has been removed. Removing Cygwin simplifies the Polyspace product installation process while improving the performance and robustness of the Polyspace Verification process.

Compatibility Considerations

Due to the Cygwin changes, Polyspace Client for Ada Version 5.1 is not compatible with previous versions of Polyspace products on Windows platforms. To avoid compatibility problems on Windows platforms, you must upgrade all your Polyspace client and server products at the same time.

If your Polyspace server is running on a Windows platform, the binary files used for batch commands in previous releases will not work without Cygwin software installed. In version 5.1, the software provides new .exe files for these batch commands. However, these files are now located in a different location.

Commands	Previous Location	New Location
Standard	<i>PolyspaceInstallDir</i> \verifier\bin\	<i>PolyspaceInstallDir</i> \verifier\wbin\
Remote Launcher	<i>Polyspace_Common</i> \RemoteLauncher\bin\	<i>Polyspace_Common</i> \RemoteLauncher\wbin\
Viewer	<i>Polyspace_Common</i> \Viewer\bin\	<i>Polyspace_Common</i> \Viewer\wbin\

If you wrote scripts using batch commands in previous releases, you must modify the scripts to use the new commands.

In addition, if you used Cygwin shell scripts for postprocessing or target compilation, those scripts will no longer run on version 5.1. To support scripting, the Polyspace software now includes Perl. You can access Perl in:

PolyspaceInstallDir\verifier\tools\perl\win32\bin\perl.exe

Enhanced Installer

Version 5.1 includes an enhanced and simplified installer for all Polyspace products. The installation process is now faster and easier to complete than in previous releases.

For more information, see the Polyspace Installation Guide.

Viewer Improvements

Enhanced exploring capability in the viewer to provide more focused information.

Unnecessary information has been eliminated from the Procedural Entities (RTE) View and Call Tree View to improve usability.

Enhanced Compilation Checks

Enhanced compilation checks to stop verification only when a pointer to a task is initiated or used, rather than when it is declared.

One-Click Enhancements

Enhanced Polyspace-In-One-Click options, to allow switching between multiple projects using a browse history.

Operating System Support

Added support for the following operating systems:

- Solaris 2.10
- Windows XP x64 (32-bit mode)

For more information, see the Polyspace Installation Guide.

Polyspace Server for Ada Product

Removed Cygwin Software Dependency for Windows Platforms

Compatibility Considerations: Yes

Previous versions of Polyspace products used Cygwin emulation to run UNIX commands on Windows systems.

In version 5.1, the Cygwin software dependency has been removed. Removing Cygwin simplifies the Polyspace product installation process while improving the performance and robustness of the Polyspace Verification process.

Compatibility Considerations

Due to the Cygwin changes, Polyspace Server™ for Ada Version 5.1 is not compatible with previous versions of Polyspace products on Windows platforms. To avoid compatibility problems on Windows platforms, you must upgrade all your Polyspace client and server products at the same time.

If your Polyspace server is running on a Windows platform, the binary files used for batch commands in previous releases will not work without Cygwin software installed. In version 5.1, the software provides new .exe files for these batch commands. However, these files are now located in a different location.

Commands	Previous Location	New Location
Standard	<i>PolyspaceInstallDir</i> \verifier\bin\	<i>PolyspaceInstallDir</i> \verifier\wbin\
Remote Launcher	<i>Polyspace_Common</i> \RemoteLauncher\bin\	<i>Polyspace_Common</i> \RemoteLauncher\wbin\
Viewer	<i>Polyspace_Common</i> \Viewer\bin\	<i>Polyspace_Common</i> \Viewer\wbin\

If you wrote scripts using batch commands in previous releases, you must modify the scripts to use the new commands.

In addition, if you used Cygwin shell scripts for postprocessing or target compilation, those scripts will no longer run on version 5.1. To support scripting, the Polyspace software now includes Perl. You can access Perl in:

PolyspaceInstallDir\verifier\tools\perl\win32\bin\perl.exe

Enhanced Installer

Version 5.1 includes an enhanced and simplified installer for all Polyspace products. The installation process is now faster and easier to complete than in previous releases.

For more information, see the Polyspace Installation Guide.

Operating System Support

Added support for the following operating systems:

- Solaris 2.10
- Windows XP x64 (32-bit mode)

For more information, see the Polyspace Installation Guide.