

Polyspace[®] Bug Finder[™] Release Notes



MATLAB[®]&SIMULINK[®]

How to Contact MathWorks



Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us



Phone: 508-647-7000



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

Polyspace[®] Bug Finder[™] Release Notes

© COPYRIGHT 2013–2015 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.

Simplified workflow for project setup and results review with a unified user interface	1-2
Code complexity metrics available in user interface	1-3
Context-sensitive help for code complexity metrics, MISRA-C:2012, and custom coding rules	1-3
Review of latest results compared to the last run	1-4
Search improvements in the user interface	1-4
Option to specify program termination functions	1-4
Simplified results infrastructure	1-5
Default statuses to justify results	1-5
Filters to limit display of results	1-5
Support for GCC 4.8	1-5
Improvements in coding rules checking	1-6
Polyspace plug-in for Simulink improvements	1-7
Integration with Simulink projects	1-7
Back-to-model available when Simulink is closed	1-8
Changes to Bug Finder defects	1-8
Polyspace binaries being removed	1-9

Import Visual Studio project being removed	1-9
---------------------------------------------------------	------------

R2014b

Support for MISRA C:2012	2-2
Parallel compilation for faster analysis	2-2
Additional concurrency issue detection (deadlocks, double locks, and others)	2-2
Data race errors	2-2
Locking errors	2-3
Support for Mac OS	2-3
Support for C++11	2-4
Context-sensitive help for analysis options and defects	2-4
Code editor in Polyspace interface	2-5
New and updated defect checkers	2-5
Ignore files and folders during analysis	2-6
Simulink plug-in support for custom project files	2-6
TargetLink support updated	2-7
AUTOSAR support added	2-7
Remote launcher and queue manager renamed	2-7
Improved global menu in user interface	2-8
Improved Project Manager perspective	2-8
Improved Results Manager perspective	2-9

Error mode removed from coding rules checking	2-9
Polyspace binaries being removed	2-9
Import Visual Studio project being removed	2-10

R2014a

Automatic project setup from build systems	3-2
Classification of bugs according to the Common Weakness Enumeration (CWE) standard	3-2
Additional coding rules support (MISRA-C:2004 Rule 18.2, MISRA-C++ Rule 5-0-11)	3-3
Support for GNU 4.7 and Microsoft Visual Studio C++ 2012 dialects	3-3
Simplification of coding rules checking	3-3
Preferences file moved	3-5
Security level support for batch analysis	3-5
Interactive mode for remote analysis	3-5
Default text editor	3-6
Results folder appearance in Project Browser	3-6
Results manager improvements	3-8
Support for Windows 8 and Windows Server 2012	3-9
Function replacement in Simulink plug-in	3-9
Check model configuration automatically before analysis .	3-10

Additional back-to-model support for Simulink plug-in . . .	3-10
Additional analysis checkers	3-11
Data range specification support	3-11
Polyspace binaries being removed	3-11
Improvement of floating point precision	3-11

R2013b

Introduction of Polyspace Bug Finder	4-2
Detection of run-time errors, data flow problems, and other defects in C and C++ code	4-2
Fast analysis of large code bases	4-2
Compliance checking for MISRA-C:2004, MISRA-C++:2008, JSF++, and custom naming conventions	4-3
Cyclomatic complexity and other code metrics	4-3
Eclipse integration	4-3
Traceability of code verification results to Simulink models	4-3
Access to Polyspace Code Prover results	4-4

R2015a

Version: 1.3

New Features

Bug Fixes

Compatibility Considerations

Simplified workflow for project setup and results review with a unified user interface

In R2015a, the Project and Results Manager perspectives have been unified. You can run verification and review results without switching between two perspectives.

The unification has resulted in the following major changes:

- After a verification, the result opens automatically.

Previously, after a verification, you had to double-click the result in the **Project Browser** to open your new results.

- You can have any of the panes open in the unified interface.

Previously, you could open the following panes only in one of the two perspectives.

Project Manager	Results Manager
<ul style="list-style-type: none"> • Project Browser: Set up project. • Configuration: Specify analysis options for your project. • Output Summary: Monitor progress of verification. • Run Log: Find information about a verification. 	<ul style="list-style-type: none"> • Results Summary: View Polyspace® results. • Source: View read-only form of source code color coded with Polyspace results. • Check Details: View details of a particular result. • Results Properties: Same as Run Log, but associated with results instead of a project. This pane has been removed. <p>To open the log associated with a result, with the results open, select Window > Show/Hide View > Run Log.</p> <ul style="list-style-type: none"> • Settings: Same information as Configuration, but associated with results instead of a project. This pane has been removed.

Project Manager	Results Manager
	To open the configuration associated with a result, with the results open, select Window > Show/Hide View > Configuration .

Code complexity metrics available in user interface

In R2015a, code complexity metrics can be viewed in the Polyspace user interface. For more information, see “Code Metrics”. Previously, this information was available only in the Polyspace Metrics web interface.

In the user interface, you can:

- Specify a limit for the value of a metric. If the metric value for your source exceeds this limit, the metric appears red in **Results Summary**.
- Comment and justify the value of a metric. If a metric value exceeds specified limits and appears red, you can add a comment with the rationale.

Using Polyspace results in this way, you can enforce coding standards across your organization. For more information, see “Review Code Metrics”.

Reducing the complexity of your code improves code readability, reduces the possibility of coding errors, and allows more precise Polyspace verification.

Context-sensitive help for code complexity metrics, MISRA-C:2012, and custom coding rules

In R2015a, context-sensitive help is available in the user interface for code metrics results, MISRA C[®]:2012 rule violations, and custom coding rule violations.

To access the contextual help, see “Getting Help”.

For information about these results, see:

- “Code Metrics”
- “MISRA C:2012 Directives and Rules”

- “Custom Coding Rules”

Review of latest results compared to the last run

In R2015a, you can review only new results compared to the previous run.

If you rerun your analysis, the new results are displayed with an asterisk (*) against them on the **Results Summary** pane. To display only these results, select the **New results** box.

If you make changes in your source code, you can use this feature to see only the results introduced due to those changes. You can avoid reviewing the results in your existing source code.

Search improvements in the user interface

In R2015a, the **Search** pane allows you to search for a string in various panes of the user interface.

To search for a string in the new user interface:

- 1** If the **Search** pane is not visible, open it. Select **Window > Show/Hide View > Search**.
- 2** Enter your string in the search box.
- 3** From the drop-down list beside the box, select names of panes you want to search.

The **Search** pane consolidates the previously available search options.

Option to specify program termination functions

In R2015a, you can specify functions that behave like the exit function and terminate your program.

- At the command line, use the flag `-termination-functions`.
- In the user interface, on the **Configuration** pane, select **Advanced Settings**. Enter `-termination-functions` in the **Other** field.

For more information, see `-termination-functions`.

Simplified results infrastructure

Polyspace results folders are reorganized and simplified. Files have been removed, combined, renamed, or moved. The infrastructure changes do not change the analysis results that you see in the Polyspace environment.

Some important changes and file locations:

- The main results file is now encrypted and renamed `ps_results.psbf`. You can view results only in the Polyspace environment.
- The log file, `Polyspace_R2015a_project_date-time.log` has not changed.

For more information, see “Results Folder Contents”.

Default statuses to justify results

Polyspace Bug Finder™ results use certain statuses to calculate the number of justified results in Polyspace Metrics.

In R2015a, the default statuses that mark results as justified are:

- **Justified** — Previously called **Justify**, renamed in R2015a.
- **No action planned** — Existing status added to justified list in R2015a.

You can change which statuses mark results as justified from the Polyspace preferences. For more information, see “Define Custom Review Status”.

Filters to limit display of results

In R2015a, you can use the **Show** menu on the **Results Summary** pane to suppress certain Polyspace Bug Finder results from display.

- To suppress code complexity metrics from display, select **Show > Defects & Rules**.
- Create your own options on the **Show** menu. Select **Tools > Preferences** and create new options through the **Review Scope** tab.

For more information, see “Limit Display of Defects”.

Support for GCC 4.8

Polyspace now supports the GCC 4.8 dialect for C and C++ projects.

To allow GCC 4.8 extensions in your Polyspace Bug Finder analysis, set the **Target & Compiler > Dialect** option to `gnu4.8`.

For more information, see “Dialect (C)” and “Dialect (C++)”.

Improvements in coding rules checking

MISRA C:2004 and MISRA AC AGC

Rule Number	Effect	More Information
Rule 12.6	More results on noncompliant <code>#if</code> preprocessor directives. Fewer results for variables cast to effective Boolean types.	MISRA C:2004 Rules — Chapter 12: Expressions
Rule 12.12	Fewer results when converting to an array of <code>float</code>	MISRA C:2004 Rules — Chapter 12: Expressions

MISRA C:2012

Rule Number	Effect	More Information
Rules 10.3	Fewer results on enumeration constants when the type of the constant is a named enumeration type. Fewer results on user-defined effective Boolean types.	MISRA C:2012 Rule 10.3
Rule 10.4	Fewer results on enumeration constants when the type of the constant is a named enumeration type. Fewer results for casts to user-defined effective Boolean types.	MISRA C:2012 Rule 10.4
Rule 10.5	Fewer results on enumeration constants when the type of the constant is a named enumeration type. Fewer results on user-defined effective Boolean types.	MISRA C:2012 Rule 10.5

Rule Number	Effect	More Information
Rule 12.1	More results on expressions with <code>sizeof</code> operator and on expressions with <code>?</code> operators. Fewer results on operators of the same precedence and in preprocessing directives.	MISRA C:2012 Rule 12.1
Rule 14.3	No results for non-controlling expressions.	MISRA C:2012 Rule 14.3

MISRA C++:2008

Rule Number	Effect	More Information
Rule 5-0-3	Fewer results on enumeration constants when the type of the constant is the enumeration type.	MISRA C++ Rules — Chapter 5
Rule 6-5-1	Fewer results on compliant vector variable iterators.	MISRA C++ Rules — Chapter 6
Rule 14-8-2	Fewer results for functions contained in the “Files and folders to ignore (C++)” option.	MISRA C++ Rules — Chapter 14
Rule 15-3-2	Fewer results for user-defined return statements after a <code>try</code> block.	MISRA C++ Rules — Chapter 15

Polyspace plug-in for Simulink improvements

In R2015a, there are three improvements to the Polyspace Simulink[®] plug-in.

Integration with Simulink projects

You can now save your Polyspace results to a Simulink project. Using this feature, you can organize and control your Polyspace results alongside your model files and folders.

To save your results to a Simulink project:

- 1 Open your Simulink project.
- 2 From your model, select **Code > Polyspace > Options**.

- 3** In the Polyspace parameter configuration tab, select the **Save results to Simulink project** option.

For more information, see “Save Results to a Simulink Project”.

Back-to-model available when Simulink is closed

In the Polyspace plug-in for Simulink, the back-to-model feature now works even when your model is closed. When you click a link in your Polyspace results, MATLAB® opens your model and highlights the related block.

Note: This feature works only with Simulink R2013b and later.

For more information about the back-to-model feature, see “Review Generated Code Results”.

Changes to Bug Finder defects

Defect	R2015a change
Invalid use of floating point operation	Off by default.
Line with more than one statement	Off by default.
Invalid use of = (assignment) operator	On by default for handwritten code (analyses started at the command-line or Polyspace environment). Off by default for generated code (analyses started from the Simulink plug-in).
Invalid use of == (equality) operator	On by default for handwritten code. Off by default for generated code.
Missing null in string array	On by default for handwritten code. Off by default for generated code.
Partially accessed array	On by default for handwritten code. Off by default for generated code.

Defect	R2015a change
Variable shadowing	On by default for handwritten code. Off by default for generated code.
Write without further read	On by default for handwritten code. Off by default for generated code.
Wrong type used in sizeof	On by default for handwritten code. Off by default for generated code.

Polyspace binaries being removed

The following binaries will be removed in a future release. The binaries to use are located in *matlabroot/polyspace/bin*. You get a warning if you run them.

Binary name	Use instead
<code>polyspace-r1-manager.exe</code>	<code>polyspace-server-settings.exe</code>
<code>polyspace-spooler.exe</code>	<code>polyspace-job-monitor.exe</code>
<code>polyspace-ver.exe</code>	<code>polyspace-bug-finder-nodesktop -ver</code>

Import Visual Studio project being removed

The **Tools > Import Visual Studio project** will be removed in a future release. Instead, use the **Create from build system** option during new project creation. For more information, see “Create Project Automatically”.

R2014b

Version: 1.2

New Features

Bug Fixes

Compatibility Considerations

Support for MISRA C:2012

Polyspace can now check your code against MISRA C:2012 directives and coding rules. To check for MISRA C:2012 coding rule violations:

- 1 On the **Configuration** pane, select **Coding Rules**.
- 2 Select **Check MISRA C:2012**.
- 3 The MISRA C:2012 guidelines have different categories for handwritten and automatically generated code.

If you want to use the settings for automatically generated code, also select **Use generated code requirements**.

For more information about supported rules, see MISRA C:2012 Coding Directives and Rules.

Parallel compilation for faster analysis

Starting in R2014b, Polyspace Bug Finder can run the compilation phase of your analysis in parallel on multiple processors. The software detects available processors and uses them to compile different source files in parallel.

Previously, the software ran post-compilation phases in parallel but compiled the source files sequentially. Starting in R2014b, the software can use multiple processors for the entire analysis process.

To explicitly specify the number of processors, use the command-line option `-max-processes`. For more information, see `-max-processes`.

Additional concurrency issue detection (deadlocks, double locks, and others)

Data race errors

The following defects deal with unprotected access of shared variables by multiple tasks.

Defect name	Status	More information
Race conditions	Removed	Replaced by Data race and Data race including atomic operations.

Defect name	Status	More information
Data race	New	Checks for unprotected operations on variables shared by multiple tasks. This check applies to non-atomic operations only.
Data race including atomic operations	New	Checks for unprotected operations on variables shared by multiple tasks. This check applies to all operations, including atomic ones.

Locking errors

The following defects deal with incorrect design of critical sections. For multitasking analysis, to mark a section of code as a critical section, you must place it between two function calls. A lock function begins a critical section. An unlock function ends a critical section.

Defect name	Status	More information
Deadlock	New	Checks whether the sequence of calls to lock functions is such that two tasks block each other.
Missing lock	New	Checks whether an unlock function has a corresponding lock function.
Missing unlock	New	Checks whether a lock function has a corresponding unlock function.
Double lock	New	Checks whether a lock function is called twice in a task without an unlock function being called in between.
Double unlock	New	Checks whether an unlock function is called twice in a task without a lock function being called in between.

For more information, see:

- Set Up Multitasking Analysis
- Review Concurrency Defects

Support for Mac OS

You can install and run Polyspace on Mac OS X. Polyspace is supported for Mac OS 10.7.4+, 10.8, and 10.9.

You can use Polyspace Metrics on Safari and set up your Mac as a Metrics server. However, if you restart your Mac machine that is setup as a Metrics server, you must restart the Polyspace server daemon.

Support for C++11

Polyspace can now fully analyze C++ code that follows the ISO[®]/IEC 14882:2011 standard, also called C++11.

Use two new analysis options when analyzing C++11 code. On the **Target & Compiler** pane, select:

- **C++11 extensions** to allow the standard C++11 libraries and functions during your analysis.
- **Block char 16/32_t types** to not allow char16_t or char32_t types during the analysis.

For more information, see C++11 Extensions (C++) and Block char16/32_t types (C++).


Context-sensitive help for analysis options and defects

Contextual help is available for analysis options in the Polyspace interface and its plugins. To view the contextual help for analysis options:

- 1 Hover your cursor over an analysis option in the **Configuration** pane.
- 2 Inside the tooltip, select the “More Help” link.

The documentation for that analysis option appears in a dockable window.

Contextual help is available for defects in the Polyspace interface. To view the contextual help:

- 1 In the Results Manager perspective, select a defect from the Results Summary.
- 2 Inside the **Check Details** pane, select .

The documentation for that Bug Finder defect appears in a dockable window.

For more information, see Getting Help.

Code editor in Polyspace interface

In R2014b, you can edit your source files inside the Polyspace user interface.

- In the Project Manager perspective, on the **Project Browser** tree, double-click your source file.
- In the Results Manager perspective, right-click the **Source** pane and select **Open Source File**.

Your source files appear on a **Code Editor** tab. On this tab, you can edit your source files and save them.

New and updated defect checkers

Defect name	Status	More information
Dead code	Updated	Checks for non-executed code. No longer checks for: <ul style="list-style-type: none">• <code>if</code> conditions that are always true without a corresponding <code>else</code>. This check is covered by the Useless if defect.• Code following control-flow statements such as <code>break</code>, <code>return</code>, or <code>goto</code> defect. This check is covered by the Unreachable code defect.
Useless if	New	Checks for if-conditions that are always true.
Unreachable code	New	Checks for code following control-flow statements such as <code>break</code> , <code>return</code> , or <code>goto</code> .
Declaration mismatch	Updated	Updated for <code>#pragma</code> packing statements.
Race conditions	Removed	Replaced by Data race and Data race including atomic operations.
Data race	New	Checks for unprotected operations on variables shared by multiple tasks. This check applies to non-atomic operations only.
Data race including atomic operations	New	Checks for unprotected operations on variables shared by multiple tasks. This check applies to all accesses, including atomic ones.

Defect name	Status	More information
Deadlock	New	Checks whether the sequence of calls to lock functions is such that two tasks block each other.
Missing lock	New	Checks whether an unlock function has a corresponding lock function.
Missing unlock	New	Checks whether a lock function has a corresponding unlock function.
Double lock	New	Checks whether a lock function is called twice in a task without an unlock function being called in between.
Double unlock	New	Checks whether an unlock function is called twice in a task without a lock function being called in between.

Ignore files and folders during analysis

You can now use the analysis option **Files and folders to ignore** (command line - `includes-to-ignore`) to ignore files and folders during defect checking. Previously, the **Files and folders to ignore** option (command line - `includes-to-ignore`) ignored files and folders during coding rule checking. In R2014b, Polyspace Bug Finder uses this option to ignore specified files or folders for coding rule checking AND defect analysis.

For more information, see `Files and folders to ignore (C)` or `Files and folders to ignore (C++)`.

Simulink plug-in support for custom project files

With the Polyspace plug-in for Simulink, you can now use a project file to specify the verification options.

On the **Polyspace** pane of the Configuration Parameters window, with the **Use custom project file** option you can enter a path or browse for a `.psprj` project file.

For more information, see `Configure Polyspace Analysis Options`.

TargetLink support updated

The Polyspace plug-in for Simulink now supports TargetLink® 3.4 and 3.5. Older versions of TargetLink are no longer supported.

For more information, see TargetLink Considerations.

AUTOSAR support added

In R2013b, the Polyspace plug-in for Simulink added support for AUTOSAR generated code with Embedded Coder®. If you use `autosar.tlc` as your **System target file** for code generation, Polyspace can analyze this generated code. Polyspace uses the same default analysis options and parameters as Embedded Coder.

For more information, see Embedded Coder Considerations.

Remote launcher and queue manager renamed

Polyspace renamed the remote launcher and the queue manager.

Previous name	New name	More information
<code>polyspace-rl-manager</code>	<code>polyspace-server-settings</code>	Only the binary name has changed. The interface title, Metrics and Remote Server Settings , is unchanged.
<code>polyspace-spooler</code>	<code>polyspace-job-monitor</code>	The binary and the interface titles have changed. Interface labels have changed in the Polyspace interface and its plug-ins.
Queue Manager or Spooler	Job Monitor	
<code>pslinkfun('queuemanager')</code>	<code>pslinkfun('jobmonitor')</code>	See <code>pslinkfun</code>

Compatibility Considerations

If you use the old binaries or functions, you receive a warning.

Improved global menu in user interface

The global menu in the Polyspace user interface has been updated. The following table lists the current location for the existing global menu options.

Goal	Prior to R2014b	R2014b
Open the Polyspace Metrics interface in your web browser.	File > Open Metrics Web Interface	Metrics > Open Metrics
Upload results from the Polyspace user interface to Polyspace Metrics.	File > Upload in Polyspace Metrics repository	Metrics > Upload to Metrics
Update results stored in Polyspace Metrics with your review comments and justifications.	File > Save in Polyspace Metrics repository	Metrics > Save comments to Metrics
Generate a report from results after verification.	Run > Run Report > Run Report	Reporting > Run Report
Open a generated report.	Run > Run Report > Open Report	Reporting > Open Report
Import review comments from a previous verification.	Review > Import	Tools > Import Comments
Specify code generator for generated code.	Review > Code Generator Support	Tools > Code Generator Support
Specify settings that apply to every Polyspace project.	Options > Preferences	Tools > Preferences
Specify settings for remote verification.	Options > Metrics and Remote Server Settings	Metrics > Metrics and Remote Server Settings

Improved Project Manager perspective

The following changes have been made in the Project Manager perspective:

- The **Progress Monitor** tab does not exist anymore. Instead, after you start a verification, you can view its progress on the **Output Summary** tab.
- In the **Project Browser**, projects appear sorted in alphabetical order instead of order of creation.

-
- On the **Configuration** pane, the **Interactive** option has been removed from the graphical interface. To use the interactive mode, use the `-interactive` flag at the command line, or in the **Advanced Settings > Other** text field. For more information, see `-interactive`

Improved Results Manager perspective

The following changes have been made in the Results Manager perspective:

- To group your defects, use the **Group by** menu on the **Results Summary** pane.
 - To leave your defects ungrouped, instead of **List of Checks**, select **Group by > None**.
 - To group defects by category, instead of **Checks by Family**, select **Group by > Family**.
 - To group defects by file and function, instead of **Checks by File/Function**, select **Group by > File**.
- On the **Source** pane:
 - If a color appears on a brace enclosing a code block, double-click the brace to highlight the block. If no color appears, click the brace once to highlight the code block.
 - If a code block is deactivated due to conditional compilation, it appears gray.

Error mode removed from coding rules checking

In R2014b, the **Error** mode has been removed from coding rules checking. Therefore, coding rule violations cannot stop a verification.

Compatibility Considerations

For existing coding rules files, coding rules that use the keyword `error` are treated in the same way as that with keyword `warning`. For more information on `warning`, see [Format of Custom Coding Rules File](#).

Polyspace binaries being removed

The following binaries will be removed in a future release. Unless otherwise noted, the binaries to use are located in `MATLAB Install/polyspace/bin`.

Binary name	What happens	Use instead
polyspace-rl-manager.exe	Warning	polyspace-server-settings.exe
polyspace-spooler.exe	Warning	polyspace-job-monitor.exe
polyspace-ver.exe	Warning	polyspace-bug-finder-nodesktop -ver
setup-remote-launcher.exe	Warning	<i>MATLAB install</i> /toolbox/polyspace / psdistcomp/bin/setup-polyspace-cluster

Import Visual Studio project being removed

The **File > Import Visual Studio project** will be removed in a future release. Instead, use the **Create from build system** option during New Project creation. For more information, see [Create Projects Automatically from Your Build System](#).

R2014a

Version: 1.1

New Features

Bug Fixes

Compatibility Considerations


Automatic project setup from build systems

In R2014a, you can set up a Polyspace project from build automation scripts that you use to build your software application. The automatic project setup runs your automation scripts to determine:

- Source files
- Includes
- **Target & Compiler** options

To set up a project from your build automation scripts:

- At the command line: Use the `polyspace-configure` command. For more information, see [Create Project from DOS and UNIX Command Line](#).
- In the user interface: When creating a new project, in the Project – Properties window, select **Create from build command**. In the following window, enter:
 - The build command that you use.
 - The folder from which you run your build command.
 - Additional options. For more information, see [Create Project in User Interface](#).

Click . In the **Project Browser**, you see your new Polyspace project with the required source files, include folders, and **Target & Compiler** options.

- On the MATLAB command line: Use the `polyspaceConfigure` function. For more information, see [Create Project from MATLAB Command Line](#).

Classification of bugs according to the Common Weakness Enumeration (CWE) standard

In R2014a, Polyspace Bug Finder associates CWE™ IDs with many defects. For the covered defects, the IDs are listed in the **CWE ID** column on the **Results Summary** pane. To view the **CWE ID** column, right-click the **Results Summary** tab and select the **CWE ID** column.

For more information, see [Common Weakness Enumeration from Bug Finder Defects](#).

Additional coding rules support (MISRA-C:2004 Rule 18.2, MISRA-C++ Rule 5-0-11)

The Polyspace coding rules checker now supports two additional coding rules: MISRA C 18.2 and MISRA[®] C++ 5-0-11.

- MISRA C 18.2 is a required rule that checks for assignments to overlapping objects.
- MISRA C++ 5-0-11 is a required rule that checks for the use of the plain `char` type as anything other than storage or character values.
- MISRA C++ 5-0-12 is a required rule that checks for the use of the signed and unsigned `char` types as anything other than numerical values.

For more information, see MISRA C:2004 Coding Rules or MISRA C++ Coding Rules.

Support for GNU 4.7 and Microsoft Visual Studio C++ 2012 dialects

Polyspace supports two additional dialects: Microsoft[®] Visual Studio[®] C++ 2012 and GNU[®] 4.7. If your code uses language extensions from these dialects, specify the corresponding analysis option in your configuration. From the **Target & Compiler > Dialect** menu, select:

- `gnu4.7` for GNU 4.7
- `visual11.0` for Microsoft Visual Studio C++ 2012

For more information, see Dialects for C or Dialects for C++.

Simplification of coding rules checking

In R2014a, the **Error** mode has been removed from coding rules checking. This mode applied only to:

- The option **Custom** for:
 - **Check MISRA C rules**
 - **Check MISRA AC AGC rules**
 - **Check MISRA C++ rules**
 - **Check JSF C++ rules**
- **Check custom rules**

The following table lists the changes that appear in coding rules checking.

Coding Rules Feature	R2013b	R2014a
New file wizard for custom coding rules.	<p>For each coding rule, you can select three results:</p> <ul style="list-style-type: none"> • Error: Analysis stops if the rule is violated. <p>The rule violation is displayed on the Output Summary tab in the Project Manager perspective.</p> <ul style="list-style-type: none"> • Warning: Analysis continues even if the rule is violated. <p>The rule violation is displayed on the Results Summary pane in the Result Manager perspective.</p> <ul style="list-style-type: none"> • Off: Polyspace does not check for violation of the rule. 	<p>For each coding rule, you can select two results:</p> <ul style="list-style-type: none"> • On: Analysis continues even if the rule is violated. <p>The rule violation is displayed on the Results Summary pane in the Result Manager perspective.</p> <ul style="list-style-type: none"> • Off: Polyspace does not check for violation of the rule.
Format of the custom coding rules file.	<p>Each line in the file must have the syntax:</p> <pre>rule off error warning #comments</pre> <p>For example:</p> <pre># MISRA configuration - Proj1 10.5 off #don't check 10.5 17.2 error 17.3 warning</pre>	<p>Each line in the file must have the syntax:</p> <pre>rule off warning #comments</pre> <p>For example:</p> <pre># MISRA configuration - Proj1 10.5 off #don't check 10.5 17.2 warning 17.3 warning</pre>

Compatibility Considerations

For existing coding rules files that use the keyword **error**:

- If you run analysis from the user interface, it will be treated in the same way as the keyword **warning**. The verification will not stop even if the rule is violated. The rule violation will however be reported on the **Results Summary** pane.

- If you run analysis from the command line, the verification will stop if the rule is violated.

Preferences file moved

In R2014a, the location of the Polyspace preferences file has been changed.

Operating System	Location before R2014a	Location in R2014a
Windows®	%APPDATA%\Polyspace	%APPDATA%\MathWorks\MATLAB\R2014a\Polyspace
Linux®	/home/\$USER/.polyspace	/home/\$USER/.matlab/\$RELEASE/Polyspace

For more information, see Storage of Polyspace Preferences.

Security level support for batch analysis

When creating an MDCS server for Polyspace batch analyses, you can now add additional security levels through the **MATLAB Admin Center**. Using the **Metrics and Remote Server Settings**, the MDCS server is automatically set to security level zero. If you want additional security for your server, use the **Admin Center** button. The additional security levels require authentication by user name, cluster user name and password, or network user name and password.

For more information, see Set MJS Cluster Security.

Interactive mode for remote analysis

In R2014a, you can select an additional **Interactive** mode for remote analysis. In this mode, when you run Polyspace Bug Finder on a cluster, your local computer is tethered to the cluster through Parallel Computing Toolbox™ and MATLAB Distributed Computing Server™.

- In the user interface: On the **Configuration** pane, under **Distributed Computing**, select **Interactive**.
- On the DOS or UNIX® command line, append `-interactive` to the `polyspace-bug-finder-nodesktop` command.

- On the MATLAB command line, add the argument `'-interactive'` to the `polyspaceBugFinder` function.

For more information, see [Interactive](#).

Default text editor

In R2014a, Polyspace uses a default text editor for opening source files. The editor is:

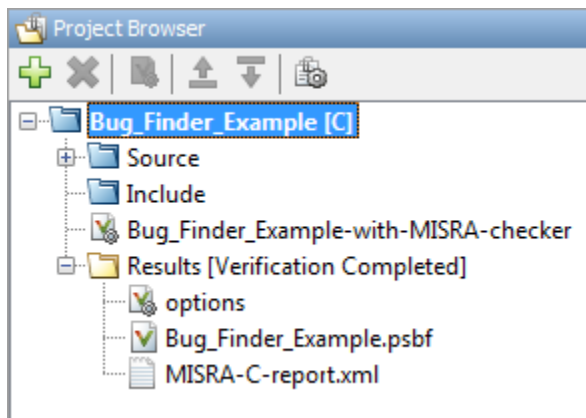
- WordPad in Windows
- `vi` in Linux

You can change the text editor on the **Editors** tab under **Options > Preferences**. For more information, see [Specify Text Editor](#).

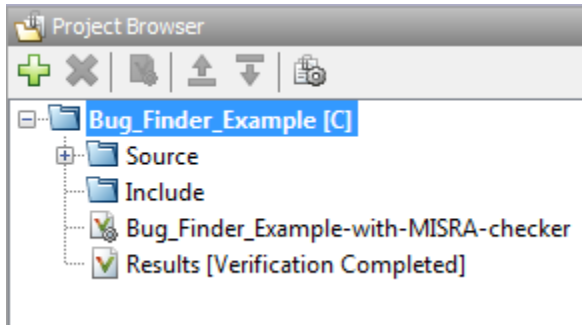
Results folder appearance in Project Browser

In R2014a, the results folder appears in a simplified form in the **Project Browser**. Instead of a folder containing several files, the result appears as a single file.

- Format before R2014a



- Format in R2014a

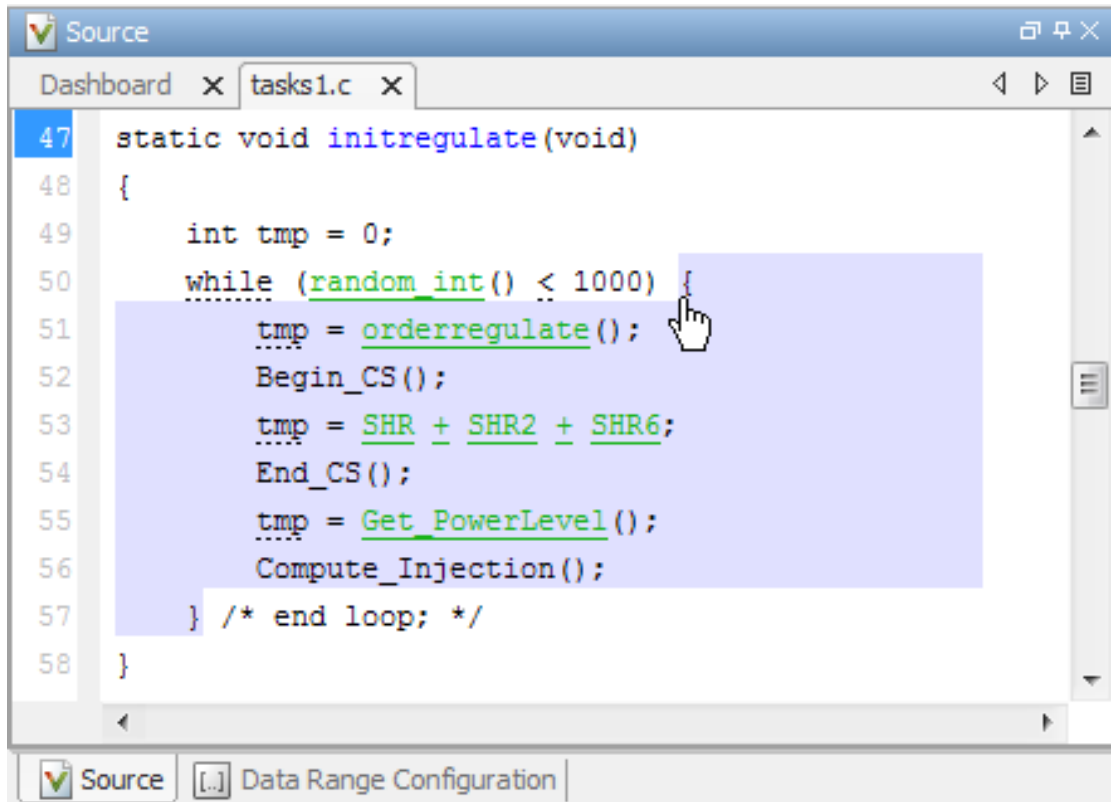


The following table lists the changes in the actions that you can perform on the results folder.

Action	R2013b	R2014a
Open results.	In the result folder, double-click result file with extension .psbf .	Double-click result file.
Open analysis options used for result.	In the result folder, select options .	Right-click result file and select Open Configuration .
Open metrics page for batch analyses if you had used the analysis option Distributed Computing > Add to results repository .	In the result folder, select Metrics Web Page .	Double-click result file. If you had used the option Distributed Computing > Add to results repository , double-clicking the results file for the first time opens the metrics web page instead of the Result Manager perspective.
Open results folder in your file browser.	Navigate to results folder. To find results folder location, select Options > Preferences . View result folder location on the Project and Results Folder tab.	Right-click result file and select Open Folder with File Manager .

Results manager improvements

- In R2014a, you can view the extent of a code block on the **Source** pane by clicking either its opening or closing brace.



Note: This action does not highlight the code block if the brace itself is already highlighted. The opening brace can be highlighted, for example, with a **Dead code** defect for the code block.

- In R2014a, the **Verification Statistics** pane in the Project Manager and the **Results Statistics** pane in the Results Manager have been renamed **Dashboard**.

On the **Dashboard**, you can obtain an overview of the results in a graphical format. You can see:

- Code covered by analysis.
- Defect distribution. You can choose to view the distribution by:
 - **File**
 - **Category** or defect name.
- Distribution of coding rule violations. You can choose to view the distribution by:
 - **File**
 - **Category** or rule number.

The **Dashboard** displays violations of different types of rules such as MISRA C, JSF[®] C++, or custom rules on different graphs.

For more information, see Dashboard.

- In R2014a, on the **Results Summary** pane, you can distinguish between violations of predefined coding rules such as MISRA C or C++ and custom coding rules.
 - The predefined rules are indicated by ▼.
 - The custom rules are indicated by ▼.

In addition, when you click the **Check** column header on the **Results Summary** pane, the rules are sorted by rule number instead of alphabetically.

- In R2014a, you can double-click a variable name on the **Source** pane to highlight other instances of the variable.

Support for Windows 8 and Windows Server 2012

Polyspace supports installation and analysis on Windows Server[®] 2012 and Windows 8.

For installation instructions, see Installation, Licensing, and Activation.

Function replacement in Simulink plug-in

The following functions have been replaced in the Simulink plug-in by the function `pslinkfun`. These functions will be removed in a future release.

Function	What Happens?	Use This Function Instead
PolyspaceAnnotation	Warning	pslinkfun('annotations',...)
PolySpaceGetTemplateCFGFile	Warning	pslinkfun('gettemplate')
PolySpaceHelp	Warning	pslinkfun('help')
PolySpaceEnableCOMServer	Warning	pslinkfun('enablebacktomodel')
PolySpaceSpooler	Warning	pslinkfun('queuemanager')
PolySpaceViewer	Warning	pslinkfun('openresults',...)
PolySpaceSetTemplateCFGFile	Warning	pslinkfun('settemplate',...)
PolySpaceConfigure	Warning	pslinkfun('advancedoptions')
PolySpaceKillAnalysis	Warning	pslinkfun('stop')
PolySpaceMetrics	Warning	pslinkfun('metrics')

For more information, see `pslinkfun`

Check model configuration automatically before analysis

For the Polyspace Simulink plug-in, the **Check configuration** feature has been enhanced to automatically check your model configuration before analysis. In the **Polyspace** pane of the Model Configuration options, select:

- **On, proceed with warnings** to automatically check the configuration before analysis and continue with analysis when only warnings are found.
- **On, stop for warnings** to automatically check the configuration before analysis and stop if warnings are found.
- **Off** does not check the configuration before an analysis.

If the configuration check finds errors, Polyspace stops the analysis.

For more information about **Check configuration**, see [Check Simulink Model Settings](#).

Additional back-to-model support for Simulink plug-in

In R2014a, the back-to-model feature is more stable. Additionally, support has been added for Stateflow[®] charts in Target Link and Linux operating systems.

For more information, see [Identify Errors in Simulink Models](#).

Additional analysis checkers

Polyspace Bug Finder can now check for two additional defects in C and C++:

- **Wrong allocated object size for cast** checks for memory allocations that are not multiples of the pointer size.
- **Line with more than one statement** checks for lines that have additional statements after a semicolon.

For more information, see [Wrong allocated object size for cast](#) and [Line with more than one statement](#).

Data range specification support

Data range specification (DRS) is available with Polyspace Bug Finder. You can add range information to global variables.

You can also use DRS information with Polyspace Code Prover™. Similarly, you can use DRS information from Code Prover in Bug Finder.

For more information, see [Inputs & Stubbing](#).

Polyspace binaries being removed

The following Polyspace binaries will be removed in a future release:

- `polyspace-report-generator.exe`
- `polyspace-results-repository.exe`
- `polyspace-spooler.exe`
- `polyspace-ver.exe`

Improvement of floating point precision

In R2013b, Polyspace improved the precision of floating point representation. Previously, Polyspace represented the floating point values with intervals, as seen in the tooltips. Now, Polyspace uses a rounding method.

For example, the analysis represents `float arr = 0.1;` as,

- Pre-R2013b, `arr = [9.9999E^-2,1.0001E-1]`.
- Now, `arr = 0.1`.

R2013b

Version: 1.0

New Features

Introduction of Polyspace Bug Finder

Polyspace Bug Finder is a new companion product to Polyspace Code Prover. Polyspace Bug Finder analyzes C and C++ code to find possible defects and coding rule violations. Bug Finder can run fast analyses on large code bases with low false-positive results. Polyspace Bug Finder also calculates code complexity metrics with Polyspace Metrics.

Bug Finder integrates with Simulink, Eclipse™, Visual Studio, and Rhapsody® to help you analyze code from within your development environment.

Detection of run-time errors, data flow problems, and other defects in C and C++ code

Polyspace Bug Finder uses static analysis to find various defects for C and C++ code with few false-positive results. The analysis does not require program execution, code instrumentation, or test cases.

Some categories of defects are:

- Numeric
- Programming
- Static memory
- Dynamic memory
- Data-flow

To see a list of defects you can find, see Polyspace Bug Finder Defects.

Bug Finder analysis runs quickly, so you can fix errors and rerun analysis.

For information about running analyses, see Find Bugs.

Fast analysis of large code bases

Polyspace Bug Finder uses an efficient analysis method which produces results quickly, even from large code bases. Therefore you can fix errors and rerun the analysis without having to wait. You can find more issues early on in the development process and produce better quality code overall.

Compliance checking for MISRA-C:2004, MISRA-C++:2008, JSF++, and custom naming conventions

Polyspace Bug Finder can also check for compliance with coding rules. There are four industry-defined rules you can select:

- MISRA C
- MISRA AC-AGC
- MISRA C++
- JSF C++

In addition, you can define rules to check for naming conventions.

You can run the coding rules checker separately, or at the same time as your analysis.

For more information, see [Check Coding Rules](#).

Cyclomatic complexity and other code metrics

Using Polyspace Metrics, Polyspace Bug Finder calculates various code metrics, including cyclomatic complexity. These statistics are displayed using Polyspace Metrics, an integrated Web interface. You can use these results to track code quality over time. You can also share the code metrics, allowing others to track your project's progress.

Eclipse integration

Polyspace Bug Finder comes with an Eclipse plug-in that integrates Polyspace into your development environment. You can set up options, run analyses, view results, and fix bugs in the Eclipse interface. Using the Polyspace plug-in, you can quickly find and fix bugs as you code.

For a tutorial on using the Polyspace Bug Finder plug-in, see [Find Defects from the Eclipse Plug-In](#).

Traceability of code verification results to Simulink models

For generated code from Simulink models, Polyspace analysis results link directly back to your Simulink model. You can trace defects back to the block that is causing the bug.

In the Source Code view of the Results Manager, the block names appear as links. When you select a link, the corresponding block is highlighted in Simulink.

For a tutorial on using Polyspace Bug Finder with Simulink models, see [Find Defects from Simulink](#).

Access to Polyspace Code Prover results

A Polyspace Bug Finder installation also includes the Polyspace Code Prover user interface. With only a Polyspace Bug Finder license, you cannot run local Polyspace Code Prover verifications in the Polyspace Code Prover interface. However, you can use the Polyspace Code Prover interface to review results and upload comments to Polyspace Metrics.

For more information, see the [Polyspace Code Prover Documentation](#).