

Polyspace[®] Code Prover[™]

Release Notes

How to Contact MathWorks



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



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



508-647-7000 (Phone)



508-647-7001 (Fax)



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

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

Polyspace[®] Code Prover[™] Release Notes

© 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.

R2013b

Proven absence of certain run-time errors in C and C++ code	2
Color-coding of run-time errors directly in code	2
Calculation of range information for variables, function parameters and return values	3
Identification of variables exceeding specified range limits	3
Quality metrics for tracking conformance to software quality objectives	4
Web-based dashboard providing code metrics and quality status	4
Guided review-checking process for classifying results and run-time error status	4
Graphical display of variable reads and writes	5
Comparison with R2013a Polyspace products	6

R2013b

Version: 9.0

New Features: Yes

Bug Fixes: No

Proven absence of certain run-time errors in C and C++ code

Use Polyspace® Code Prover™ to prove the absence of overflow, divide-by-zero, out-of-bounds array access, and certain other run-time errors in source code. To verify code, the software uses formal methods-based abstract interpretation techniques. The code verification is static. It does not require program execution, code instrumentation, or test cases. Before compilation and test, you can verify handwritten code, generated code, or a combination of these two types of code.

Color-coding of run-time errors directly in code

Polyspace Code Prover uses color coding to indicate the status of code elements.

- **Green** — Proved to never have a run-time error.
- **Red** — Proved to always have a run-time error.
- **Gray** — Proved to be unreachable, which can indicate a functional issue.
- **Orange** — Unproven, and can have an error.

Errors detected include:

- Overflows, underflows, divide-by-zero, and other arithmetic errors
- Out-of-bounds array access and illegally dereferenced pointers
- Always true/false statement due to dataflow propagation
- Read access operation on uninitialized data
- Dead code
- Access to null `this` pointer (C++)
- Dynamic errors related to object programming, inheritance, and exception handling (C++)
- Uninitialized class members (C++)
- Unsound type conversions

For more information, see “Interpret Results”.

Calculation of range information for variables, function parameters and return values

Polyspace Code Prover calculates and displays range information associated with, for example, variables, function parameters and return values, and operators. The displayed range information represents a superset of dynamic values, which the software computes using static methods.

For more information, see “Interpret Results”.

Identification of variables exceeding specified range limits

By default, Polyspace Code Prover performs a *robustness* verification of your code. The verification proves that the software works under all conditions. As the verification assumes that all data inputs are set to their full range, almost any operation on these inputs can produce an overflow.

To prove that your code works in normal conditions, use the Data Range Specification (DRS) feature to perform contextual verification. You can set constraints on data ranges, and verify your code within these ranges. The use of DRS can substantially reduce the number of orange checks in verification results.

You can use DRS to set constraints on:

- Global variables
- Input parameters for user-defined functions called by the main generator
- Return values for stub functions

For a global variable, if you specify the `globalassert` mode, the software generates a warning when the variable exceeds your specified range.

For more information, see “Data Range Configuration”.

Quality metrics for tracking conformance to software quality objectives

You can define a quality model with reference to coding rule violations, code complexity, and run-time errors. By observing these metrics, you can track your progress toward predefined software quality objectives as your code evolves from the first iteration to the final version.

By confirming the absence of certain run-time errors and measuring the rate of improvement in code quality, Polyspace Code Prover enables developers, testers, and project managers to produce, assess, and deliver code that is free of run-time errors.

For more information, see “Quality Metrics”.

Web-based dashboard providing code metrics and quality status

Polyspace Code Prover provides Polyspace Metrics, a Web-based dashboard for tracking submitted verification jobs, reviewing progress, and viewing the quality status of your code. Polyspace Metrics provides an integrated view of project metrics, displaying code complexity, coding rule violations, run-time errors, and other code metrics.

For more information, see “Quality Metrics”.

Guided review-checking process for classifying results and run-time error status

In the Results Manager perspective, Polyspace Code Prover provides you with several options to organize your review process.

- You can use review methodologies to specify the number and type of checks displayed on the **Results Summary** pane. With each methodology, you review only a subset of checks.

For example, if you are reviewing verification results for the first time, select **First checks to review**. The software displays all red and gray checks but only a subset of orange checks. These orange checks are the ones most likely to be run-time errors. For more information, see “Review Checks Using Predefined Methodologies”.

- You can group checks by File/Function or Check:
 - Grouping by Check classifies checks by color. Within each color, this grouping classifies checks by categories related to the origin of the check, such as Control flow, Data flow, and Numerical.
 - Grouping by File/Function classifies checks by the file where they originated. Within each file, this grouping classifies checks by functions where they originated.
 - For C++ files, you can also group checks by Class. This grouping classifies checks by the class definition where they originated.For more information, see “Organize Check Review Using Filters and Groups”.
- You can filter checks using any of the column information criteria on the **Results Summary** pane. For example, you can filter out checks that you have already justified using the filter icon on the **Justified** column header. If you have applied a filter, the column heading changes to indicate that all results are not displayed. You can also define custom filters. For more information, see “Organize Check Review Using Filters and Groups”.
- You can navigate through the **Results Summary** pane using the keyboard or UI buttons. Both means of navigation respect the grouping, filters, and methodology used to display results.

Graphical display of variable reads and writes

A Polyspace Code Prover verification generates a data dictionary with information about global variables and the read and write access operations on these variables. You can view this information through the **Variable Access** pane of the Results Manager perspective.

For more information, see “Exploring Results Manager Perspective”.

Comparison with R2013a Polyspace products

Polyspace Code Prover is a single product that replaces the following R2013a products:

- Polyspace Client™ for C/C++
- Polyspace Server™ for C/C++

Polyspace Bug Finder™, which is available with the Polyspace Code Prover, incorporates the following R2013a products:

- Polyspace Model Link™ SL
- Polyspace Model Link™ TL
- Polyspace UML Link™ RH

For a summary of differences and similarities in remote verification, results review and other features and options, expand the following:

Remote verification

Category	R2013a	R2013b
Products required	Install: <ul style="list-style-type: none"> • Polyspace Client for C/C++ on local computer • Polyspace Server for C/C++ on network computers, which are configured as Queue Manager and CPUs. 	Install: <ul style="list-style-type: none"> • MATLAB®, Polyspace Bug Finder, and Parallel Computing Toolbox™ on local computer. • MATLAB, Polyspace Bug Finder, Polyspace Code Prover, and MATLAB Distributed Computing Server™ on head node of computer cluster. For information about setting up a cluster, see “Install Products and Choose Cluster Configuration”.

Category	R2013a	R2013b
Configuring and starting services	<p>On the Polyspace Preferences > Server Configuration tab:</p> <ul style="list-style-type: none"> • Under Remote configuration, specify host computer for Queue Manager and Polyspace Metrics server and communication port. • Under Metrics configuration, specify other settings for Polyspace Metrics. 	<p>On the Polyspace Preferences > Server Configuration tab:</p> <ul style="list-style-type: none"> • Under MDCS cluster configuration, specify computer for cluster head node, which hosts the MATLAB job scheduler (MJS). The MJS replaces the R2013a Polyspace Queue Manager. • Under Metrics configuration: <ul style="list-style-type: none"> ▪ Specify host computer for Polyspace Metrics server and communication port. ▪ Specify other settings for Polyspace Metrics.
	<p>In the Remote Launcher Manager dialog box:</p> <ol style="list-style-type: none"> 1 Under Common Settings, specify Polyspace communication port, user details, and results folder for remote verifications. 2 Under Queue Manager Settings, specify Queue Manager and CPUs. 3 Under Polyspace Server Settings, specify available Polyspace products. 	<p>In the Metrics and Remote Server Settings dialog box:</p> <ol style="list-style-type: none"> 1 Under Polyspace Metrics Settings, specify user details, Polyspace communication port, and results folder for remote verifications. 2 Under Polyspace MDCS Cluster Security Settings, you see the following options with default values: <ul style="list-style-type: none"> • Start the Polyspace MDCE service — Selected. The mdce


Category	R2013a	R2013b
	<p>4 To start the Queue Manager and Polyspace Metrics service, click Start Daemon.</p>	<p>service, which is required to manage the MJS, runs on the MJS host computer and other nodes of the cluster.</p> <ul style="list-style-type: none"> • MDCE service port — 27350. • Use secure communication – Not selected. Communication is not encrypted. You may want to use secure communication. For information about MATLAB Distributed Computing Server cluster security, see “Cluster Security”. <p>3 To start the Polyspace Metrics and mdce services, click Start Daemon.</p> <p>Use the Metrics and Remote Server Settings dialog box to start and stop mdce services only if you configure the MDCS head node as the Polyspace Metrics server. Otherwise, clear the Start the Polyspace MDCE service check box, and use the MDCS Admin Center. To open the MDCS Admin Center, run:</p> <pre><i>MATLAB_Install</i>/toolbox/distcomp/bin/admincenter</pre> <p>For information about the MDCS Admin Center, see “Cluster Processes and Profiles”.</p>

Category	R2013a	R2013b
Running a remote verification	<p>In the Project Manager perspective:</p> <ol style="list-style-type: none"> 1 On the Configuration > Machine Configuration pane, select the following check boxes: <ul style="list-style-type: none"> • Send to Polyspace Server • Add to results repository — Allows viewing of results through Polyspace Metrics. 2 On the toolbar, click Run. <p>The Polyspace client performs code compilation and coding rule checking on the local, host computer. Then the Polyspace client submits the verification to the Queue Manager on your network.</p>	<p>In the Project Manager perspective:</p> <ol style="list-style-type: none"> 1 On the Configuration > Distributed Computing pane, select the Batch check box. By default, the software selects the Add to results repository, which enables the generation of Polyspace Metrics. 2 On the toolbar, click Run. <p>The Polyspace Code Prover software performs code compilation and coding rule checking on the local, host computer. Then the Parallel Computing Toolbox client submits the verification job to the MJS of the MATLAB Distributed Computing Server cluster.</p>
Managing remote verifications	<p>Use the Queue Manager to monitor and manage submitted jobs from Polyspace clients. On the Web, you can monitor jobs through Polyspace Metrics. If you have installed Polyspace Server for C/C++ on your local computer, through Polyspace Metrics, you can open the Queue Manager .</p>	<p>Use the Queue Manager to monitor and manage jobs submitted through Parallel Computing Toolbox clients.</p>
Accessing results of remote verifications	<p>When you run a verification on a Polyspace server, the Polyspace software automatically downloads the results to your local, client computer. You can view the results in the Results Manager perspective.</p>	<p>On the Web, use Polyspace Metrics to view verification results. If Polyspace Bug Finder is installed on your local computer, you can download verification results. For example, in Polyspace Metrics,</p>

Category	R2013a	R2013b
	<p>In addition, you can use the Queue Manager to download results of verifications submitted from other Polyspace clients. On the Web, use Polyspace Metrics to view verification results stored in results repository. If Polyspace Client for C/C++ is installed on your local computer, you can download verification results. For example, in Polyspace Metrics, clicking a cell value in the Run-Time Checks view opens the corresponding verification results in the Results Manager.</p>	<p>clicking a Project cell in the Runs view opens the corresponding verification results in the Results Manager.</p>

Results review

Category	R2013a	R2013b
Results Explorer	<p>Available. Allows navigation through checks by the file and function where they occur. To view, select Window > Show/Hide View > Results Explorer.</p>	<p>Removed. To navigate through checks by file and function, on Results Summary pane, from the drop-down menu, select File/Function.</p>
Filters on the Results Summary pane	<p>Filters appear as icons on the Results Summary pane. You can filter by:</p> <ul style="list-style-type: none"> • Run-time error category • Coding rules violated • Check color • Check justification • Check classification 	<p>You can filter by the information in all the columns of the Results Summary pane. In addition to existing filters, the new filtering capabilities extend to the file, function and line number where the checks appear. You can also define your own filters.</p>

Category	R2013a	R2013b
	<ul style="list-style-type: none"> • Check status 	<p>The filters appear as the  icon on each column header. To apply a filter using the information in a column:</p> <ol style="list-style-type: none"> 1 Place your cursor on the column header. The filter icon appears. 2 Click the filter icon and from the context menu, clear the All box. Select the appropriate boxes to see the corresponding checks. <p>For more information, see “Organize Check Review Using Filters and Groups”.</p>
Code Coverage Metrics	<p>In the Results Explorer view, the software displays two metrics for the project:</p> <ul style="list-style-type: none"> • unp — Number of unreachable functions as a ratio of total number of functions • cov — Percentage of elementary operations covered by verification <p>The unreachable procedures are marked gray in the Results Explorer view.</p>	<p>The new Results Statistics pane displays the code coverage metrics through the Code covered by verification column graph. To see a list of unreachable procedures, click this column graph.</p> <p>For more information, see “Results Statistics”.</p>

Other features

Product	Feature	R2013a	R2013b
Polyspace Client and Server for C/C++	Installation	Separate installation process for Polyspace products	Polyspace Code Prover software installed during MATLAB installation process.
	Project configuration	On host, for example, using Polyspace Client for C/C++ software.	On host, using Polyspace Code Prover software.
	Local verification	On host, run Polyspace Client for C/C++ verification. Review results in Results Manager.	On host, run Polyspace Code Prover verification. Review results in Results Manager.
	Export of review comments to Excel®, and Excel report generation	Supported	Not supported.
	Line command	polyspace-c ... polyspace-cpp ...	polyspace-code-prover-nodesktop ...
	Project configuration file extension	<i>project_name.cfg</i>	<i>project_name.psproj</i>
	Results file extension	<i>results_name.rte</i>	<i>results_name.pscp</i>
	Configuration > Machine Configuration pane	Available	Replaced by Configuration > Distributed Computing pane.
	Configuration > Post-Verification pane	Available	Renamed Configuration > Advanced Settings
	goto blocks	Not supported	Supported

Product	Feature	R2013a	R2013b
	Run verifications from multiple Polyspace environments	Supported	Not supported, produces a license error -4,0.
	Non-official options field	Available in Configuration > Machine Configuration pane	Renamed Other and moved to Configuration > Advanced Settings pane
Polyspace Model Link SL and TL	Default includes	Includes specific to the target specified.	Generic includes for C and C++. These includes are target independent.
	Running a verification	Code > Polyspace > Polyspace for Embedded Coder/Target Link <ul style="list-style-type: none"> • Verify Generated Code • Verify Generated Model Reference Code <p>Also right-clicking on a subsystem and selecting Polyspace > Polyspace for Embedded Coder/Target Link</p>	Code > Polyspace > Verify Code Generated for <ul style="list-style-type: none"> • Selected Subsystem • Model • Referenced Model • Selected Target Link Subsystem <p>Also right-clicking on a subsystem and selecting Polyspace > Verify Code Generated for > Selected Subsystem / Selected Target Link Subsystem</p>
	Product Mode	Not available.	Choose between Code Prover or Bug Finder depending on the type of analysis you want to run.

Product	Feature	R2013a	R2013b
	Settings	Available. Called Verification Settings from	Available. Called Settings from . Functionality the same.
	Open results	Option Open Project Manager and Results Manager opened the Polyspace Project Manager.	Option Open results automatically after verification opens Polyspace Metrics (batch verifications) or Polyspace Results Manager (local verifications).
Polyspace plug-in for Visual Studio® 2010	Support for C++11 features	Partial support.	Added support for: <ul style="list-style-type: none"> • Lambda functions • Rvalue references for <code>*this</code> and initialization of class objects by rvalues • Decltype • Auto keyword for multi-declarator auto and trailing return types • Static assert • Nullptr • Extended friend declarations • Local and unnamed types as template arguments

Options

Product	Option	R2013a	R2013b
	-code-metrics	Available. Not selected by default.	Removed. Code complexity metrics computed by default.
	-dialect	Available.	Default unchanged, but new value <code>gnu4.6</code> available for C and C++.
Polyspace Client and Server for C/C++	-max-processes	Specify through Machine Configuration > Number of processes for multiple CPU core systems or command line .	Specify from command line, or through Advanced Settings > Other .
	-allow-language-extensions	Available. Selected by default.	Removed. By default, software supports subset of common C language constructs and extended keywords defined by the C99 standard or supported by many compilers.
	-enum-type-definition	Available with three values. First value called <code>defined-by-standard</code> .	Available with three values. For C, first value renamed <code>signed-int</code> . For C++, first value renamed <code>auto-signed-int-first</code> .

Product	Option	R2013a	R2013b
Polyspace Model Link SL and TL	-scalar-overflows- behavior wrap-around	Available. Not selected by default.	Default. This option correctly identifies generated code from blocks with saturation enabled. However, this option might lead to a loss of precision. For models without saturation, you can choose to remove this option.
	-ignore-constant- overflows	Available. Not selected by default.	Default.