Polyspace® Code Prover™ Access™ Release Notes

MATLAB®

MathWorks®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

Contents

R2019b

Version: 2.1

New Features

Bug Fixes

User Authentication: Use LDAP search filters to restrict number of users to authenticate

In R2019b, if you use your organization's Lightweight Directory Access Protocol (LDAP) to authenticate users, you can filter for and load a subset of users from your LDAP database when you start Polyspace® Code Prover™ Access™. Previously, you loaded all LDAP users listed under the **LDAP base** that you specified when you started Polyspace Code Prover Access.

To filter the LDAP users, use the new **LDAP search filter** field in the Cluster Operator settings for the **User Manager** service. For more information, see "Use Your Organization LDAP".

User Management: Update list of users from LDAP database or LDIF file

In R2019b, if you remove users from your organization's Lightweight Directory Access Protocol (LDAP) database or from the Polyspace Access embedded LDAP LDIF file, you can update the list of users stored in the Polyspace Access database. Previously, users that were removed from the LDAP database or from the LDIF file were still visible in the list of users you selected when assigning findings or managing project permissions.

To update the list of users stored in the Polyspace Access database, append `/users/list/removed` to the URL that you use to "Open the Polyspace Access Web Interface". Only an **Administrator** can perform this operation. For more information, see "Manage LDAP Users in Polyspace Access".

R2019a

Version: 2.0

New Features

Project Dashboard: Track progress of code quality via Polyspace results

Summary: In R2019a, you can track the progress of the code quality of your projects using the new intuitive Polyspace Code Prover Access **DASHBOARD**. When an analysis run is uploaded to the Polyspace Access database, the dashboard updates to give a snapshot of all the available findings, including a progress trend for number of findings compared to previous runs.

⊟ 📊 **Summary**   Code-Prover_Example-Trends_pre (Code Prover)

Open Issues

Open	97
New	7
Assigned To Me	0
Unassigned	97

Code Metrics

Sub-project(s)	0
Number of Files	6
Number of Lines Without Comment	429
Cyclomatic Complexity	6
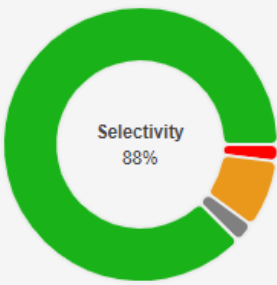
Quality Objectives

6.0%

Threshold Exhaustive ▾
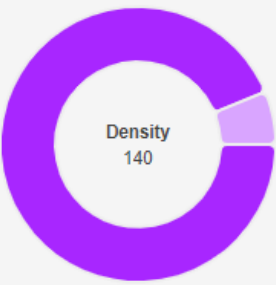
Remaining   89

Run-time Checks   Open 29

Selectivity
88%

● Red	5
● Orange	20
● Gray	6
● Green	219
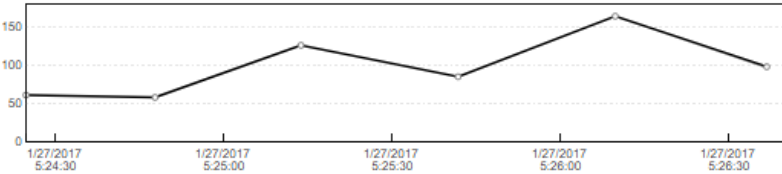
Coding Standards   Open 60

Density
140

● To Do	60
● Done	4

⊟ Trends

Number of open findings over time

1/27/2017 5:24:30	1/27/2017 5:25:00	1/27/2017 5:25:30	1/27/2017 5:26:00	1/27/2017 5:26:30

● Open

⊟ Details

Name	Total	To Do	In Progress	Done
● Red	5	4	–	1
✕ Gray	6	6	–	–
? Orange	20	19	–	1
✓ Green	219	–	–	–
▽ Coding Standards	64	60	–	4
✕ Global Variables	23	8	–	–

2-3

Additional Benefits:

- *Prioritize reviews:* See new and open issues that have not been fixed or justified, then open a detailed results list for just those issues. You can drill down on a set of findings filtered by new, open, unassigned, by family of findings, or by file.
- *Aggregate results for multiple projects:* If your team works on multiple projects, you can move all of those under an umbrella project and view a snapshot of the code quality for all your team's projects.
- *Authenticate client access:* The web interface is behind a login. Only users with a Polyspace Code Prover Access license and the appropriate credentials can view the dashboard from their web browser.

Project Dashboard: Compare Polyspace Code Prover results against Software Quality Objectives

Summary: In R2019a, check the quality of your code against pre-defined quality objective thresholds using the new Polyspace Code Prover Access **Quality Objectives** dashboard web interface. Use the thresholds to establish PASS/FAIL criteria for the code quality of your projects. For instance, the dashboard displays the progress and remaining open issues across thresholds and categories of findings. Use the available dropdown menu to select a threshold and see a more detailed view of completion by category of finding.

Summary

	Code Metrics	Coding Rules	Systematic	Dead Code	Potential
Done	97%	0%	42%	–	–
Open	12	8	4	0	0

94.0%
Threshold: SQO2
Open Issues: 24

Details

Group	SQO1	SQO2	SQO3	SQO4	SQO5	SQO6
SQO progress	95.0%	94.0%	93.0%	92.0%	92.0%	89.0%
Total Open Issues	20	24	30	34	37	49
Code Metrics	12	12	12	12	12	12
Coding Rules	8	8	8	8	10	10
Systematic	–	4	4	4	4	4
Dead Code	–	–	6	6	6	6
Potential	–	–	–	4	5	17
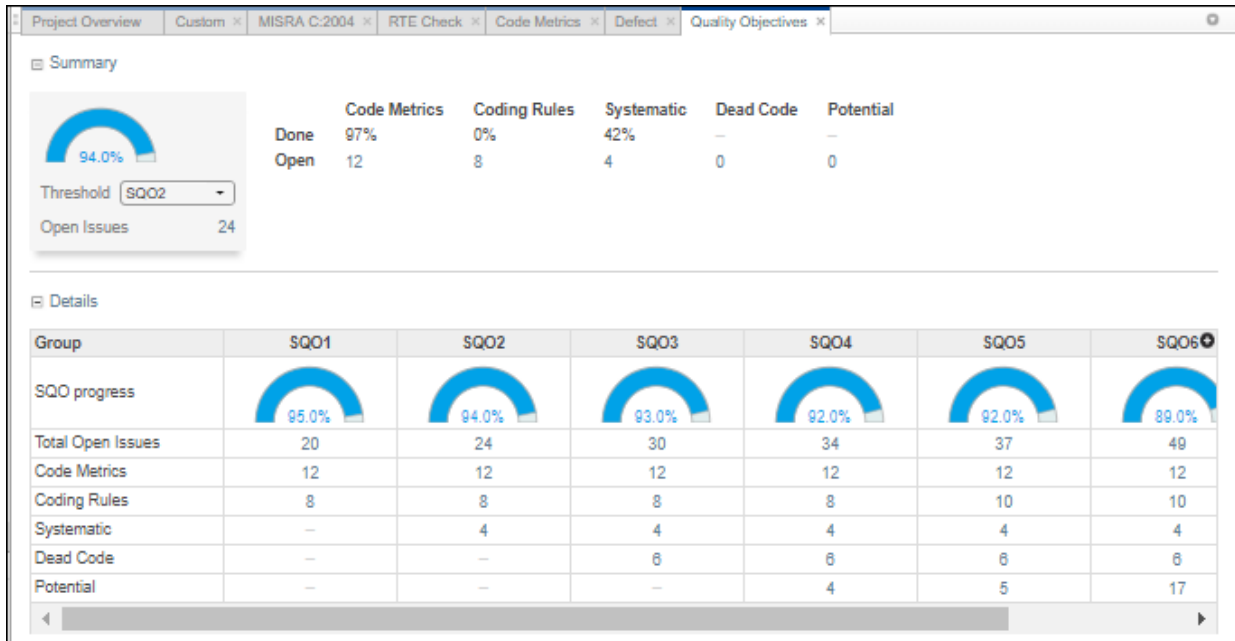
Additional Benefits:

- *Achieve better quality code:* The dashboard lets you drill down to categories of open issues for each threshold. Click a cell in the table to open a list of findings you need to address to pass a given quality objective threshold.

Collaborative Review Support: Review Polyspace Code Prover results and source code in web browser

Summary: In R2019a, review Polyspace analysis findings and view the findings in your source code using the new Polyspace Code Prover Access **REVIEW** web interface. You do not need to install a Polyspace product on your machine to open and review analysis results.

Showing: 395 / 395

Family	ID	Type	Group	Check
● *	58538	Red Check	Static memory	Illegally deref
● *	58603	Red Check	Other	Invalid use of
● *	58686	Red Check	Control flow	Non-terminat
● *	58701	Red Check	Static memory	Out of bound
● *	58845	Red Check	Control flow	Non-terminat
× *	58534	Gray Check	Data flow	Unreachable
× *	58627	Gray Check	Data flow	Unreachable
× *	58681	Gray Check	Data flow	Unreachable
× *	58725	Gray Check	Data flow	Unreachable
× *	58767	Gray Check	Data flow	Unreachable
× *	58847	Gray Check	Data flow	Unreachable
? *	58543	Orange Check	Static memory	Illegally deref
? *	58570	Orange Check	Numerical	Division by ze
? *	58582	Orange Check	Numerical	Overflow
? *	58585	Orange Check	Numerical	Overflow
? *	58589	Orange Check	Numerical	Overflow
? *	58597	Orange Check	Numerical	Overflow
? *	58599	Orange Check	Data flow	Non-initialize
? *	58601	Orange Check	Other	User assertio
? *	58626	Orange Check	Data flow	Non-initialize
? *	58674	Orange Check	Data flow	Non-initialize
? *	58675	Orange Check	Data flow	Non-initialize
? *	58676	Orange Check	Static memory	Illegally deref
? *	58707	Orange Check	Data flow	Non-initialize
? *	58712	Orange Check	Other	User assertio
? *	58766	Orange Check	Numerical	Overflow
? *	58773	Orange Check	Static memory	Out of bound
? *	58778	Orange Check	Data flow	Non-initialize
? *	58783	Orange Check	Other	User assertio
? *	58785	Orange Check	Data flow	Non-initialize
? *	58790	Orange Check	Other	User assertio
? *	58818	Orange Check	Numerical	Overflow
? *	58833	Orange Check	Numerical	Overflow
▽ *	58879	MISRA C:2012	9 Initialization	9.1 The value
▽ *	58880	MISRA C:2012	9 Initialization	9.1 The value

Result Details — example.c / Pointer_Arithmetic()

Status Unreviewed
Severity Unset
Assigned to Type username or...
Track issue Create Ticket

● Illegally dereferenced pointer ?
Error: pointer is outside its bounds
Dereference of local pointer 'p' (pointer to int 32, size: 32 bits):
 Pointer is not null.
 Points to 4 bytes at offset 400 in buffer of 400 bytes, so is outside bounds.
 Pointer may point to variable or field of variable:
  'array', local to function 'Pointer_Arithmetic'.

	Event	File	Scope
1	Entering function 'RTE'	main.c	main()
2	Entering function 'Point…	example.c	RTE()
3	● Illegally dereference…	example.c	Pointer_Arithmetic()

Source Code

example.c single_file_analysis.c

```
 94    for (i = 0; i < 100; i++) {
 95        *p = 0;
 96        p++;
 97    }
 98
 99    if (get_bus_status() > 0) {
100        if (get_oil_pressure() > 0) {
101            *p = 5; /* Out of bounds */
102        } else {
103            i++;
104        }
105    }
106
107    i = get_bus_status();
108
109    if (i >= 0)  {*(p - i) = 10;}
```

Additional Benefits:

- *Facilitate collaborative review:* The web interface streamlines the review efforts of your team. For instance:

  - During a team meeting, findings can be assessed and assigned to developers.

- Developers can log into the web interface to review findings assigned to them, and determine whether to justify the findings or fix them.
- A project manager can track the progress of the review by filtering the list of results for findings that are still open.

- *Authenticate client access:* The web interface is behind a login. Only users with a Polyspace Code Prover Access license and the appropriate credentials can view the results from their web browser.

Collaborative Review Support: Share Polyspace Code Prover results using web links

Summary: In R2019a, you can right-click an analysis result in the Polyspace Code Prover Access interface to obtain a URL that you can share with other team members. The link that you provide opens the Polyspace Code Prover Access interface and displays the finding along with the corresponding source code.

Project Authorization Management: Create and enforce authorization policies for access to project

Summary:In R2019a, you can manage project users in Polyspace Code Prover Access by right-clicking a project in the **PROJET EXPLORER** and assigning roles to member of your team. The roles authorize or forbid users from viewing projects.
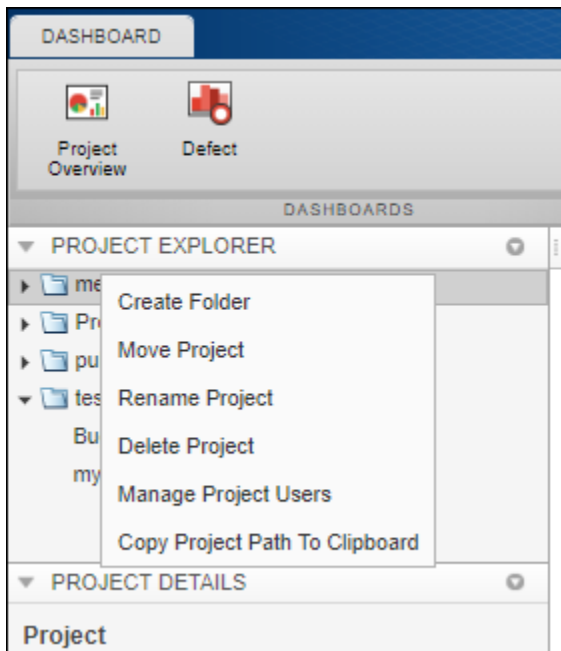

Additional Benefits:

- *Restrict access to your source code:* Use the authorization policy to restrict who can view the source code you upload with your analysis results.

- *Display relevant projects only:* When they log in to Polyspace Access, users can only see projects for which they are administrators, owners, or contributors. Use the authorization policy so that team members only see projects that they are working on.

Bug Tracking Tool Support: Create JIRA issues for Polyspace Code Prover results and assign to developer

Summary: In R2019a, Polyspace Code Prover supports integration with the JIRA software. If you have an instance of the JIRA software, after you configure Polyspace Code Prover, you can create a JIRA ticket to track Polyspace findings. The ticket is populated with details of the finding and a link to open that finding in Polyspace Access. You can add the ticket to any existing JIRA project.
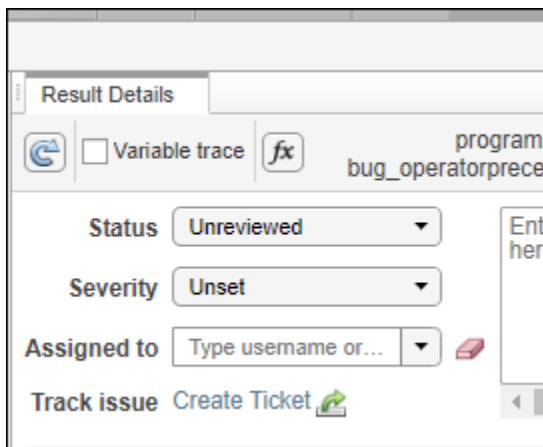
Once you create a ticket, the **Result Details** pane in the Polyspace Code Prover web interface displays a link to the corresponding JIRA issue.