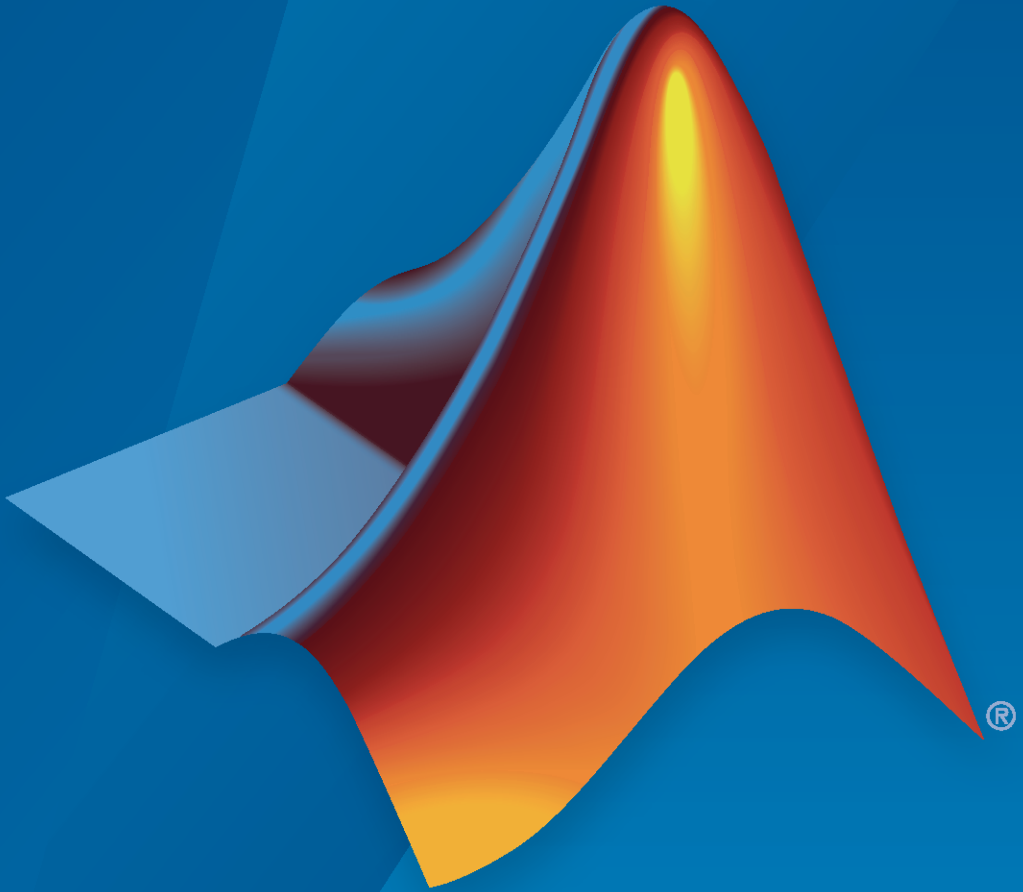
Polyspace® Bug Finder™ Access™ Release Notes



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

Polyspace® Bug Finder™ Access™ Release Notes

© COPYRIGHT 2019-2021 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.

R2021a

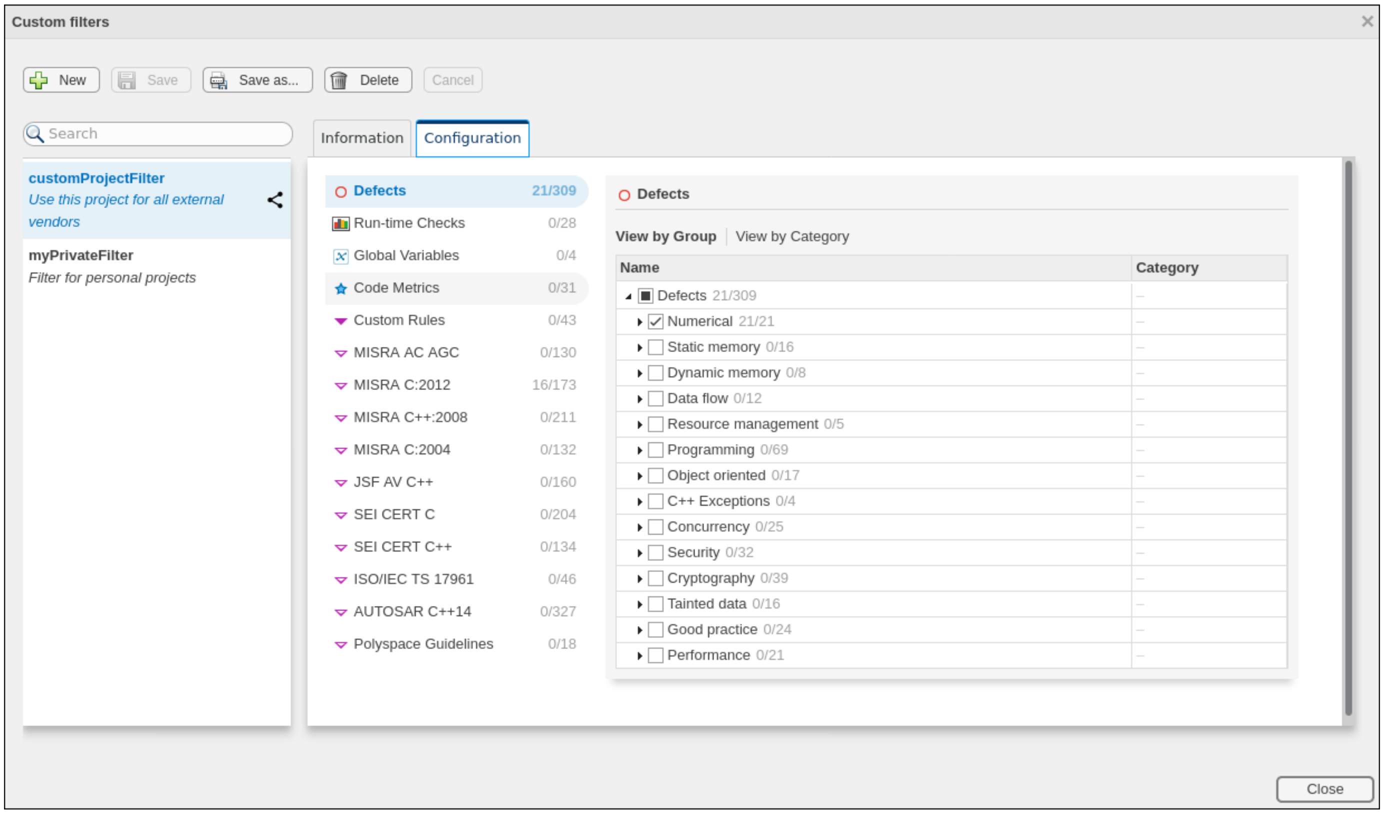
R2021a

Dashboard and Review in Web Browser

Results Review Scope: Define and share custom families of filters

In R2021a, you can create custom families of filters to tailor the scope of your review to results that are relevant to only your project or organization. You can then share the customized review scopes with other Polyspace® Access users. See “Create Custom Filter Groups in Polyspace Access Web Interface”.

For example, you might want to review your code for violations of a subset of only **Numerical** defects and **MISRA C®: 2012** rules.



customProjectFilter
Use this project for all external vendors

myPrivateFilter
Filter for personal projects

Defects	21/309
Run-time Checks	0/28
Global Variables	0/4
Code Metrics	0/31
Custom Rules	0/43
MISRA AC AGC	0/130
MISRA C:2012	16/173
MISRA C++:2008	0/211
MISRA C:2004	0/132
JSF AV C++	0/160
SEI CERT C	0/204
SEI CERT C++	0/134
ISO/IEC TS 17961	0/46
AUTOSAR C++14	0/327
Polyspace Guidelines	0/18

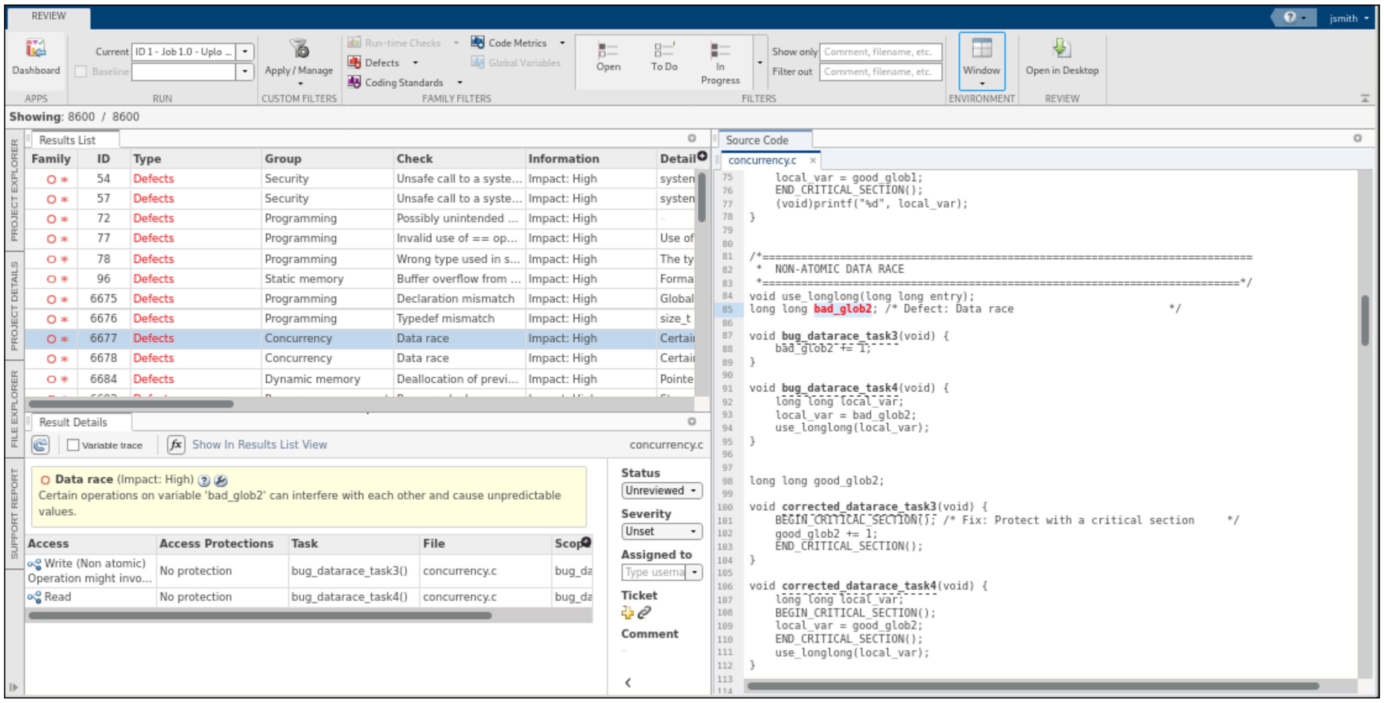
Defects

Name	Category
Defects 21/309	–
✓ Numerical 21/21	–
Static memory 0/16	–
Dynamic memory 0/8	–
Data flow 0/12	–
Resource management 0/5	–
Programming 0/69	–
Object oriented 0/17	–
C++ Exceptions 0/4	–
Concurrency 0/25	–
Security 0/32	–
Cryptography 0/39	–
Tainted data 0/16	–
Good practice 0/24	–
Performance 0/21	–

Results Review Layout: Select view to prioritize review of code or results list

In R2021a, the Polyspace Access interface has different layouts to match your results review workflow.

The default **Code Review** layout enables you to focus on the **Source Code** while you investigate issues in your code.



Current: ID 1 - Job 1.0 - Uplo ...

Show only: Comment, filename, etc.

Filter out: Comment, filename, etc.

Showing: 8600 / 8600

Family	ID	Type	Group	Check	Information	Detail
○ ✻	54	Defects	Security	Unsafe call to a syste...	Impact: High	systen
○ ✻	57	Defects	Security	Unsafe call to a syste...	Impact: High	systen
○ ✻	72	Defects	Programming	Possibly unintended ...	Impact: High	-
○ ✻	77	Defects	Programming	Invalid use of == op...	Impact: High	Use of
○ ✻	78	Defects	Programming	Wrong type used in s...	Impact: High	The ty
○ ✻	96	Defects	Static memory	Buffer overflow from ...	Impact: High	Forma
○ ✻	6675	Defects	Programming	Declaration mismatch	Impact: High	Global
○ ✻	6676	Defects	Programming	Typedef mismatch	Impact: High	size_t
○ ✻	6677	Defects	Concurrency	Data race	Impact: High	Certai
○ ✻	6678	Defects	Concurrency	Data race	Impact: High	Certai
○ ✻	6684	Defects	Dynamic memory	Deallocation of previ...	Impact: High	Pointe

concurrency.c

○ **Data race** (Impact: High)
Certain operations on variable 'bad_glob2' can interfere with each other and cause unpredictable values.

Access	Access Protections	Task	File	Scop
Write (Non atomic) Operation might invo...	No protection	bug_datarace_task3()	concurrency.c	bug_da
Read	No protection	bug_datarace_task4()	concurrency.c	bug_da

Status
Unreviewed

Severity
Unset

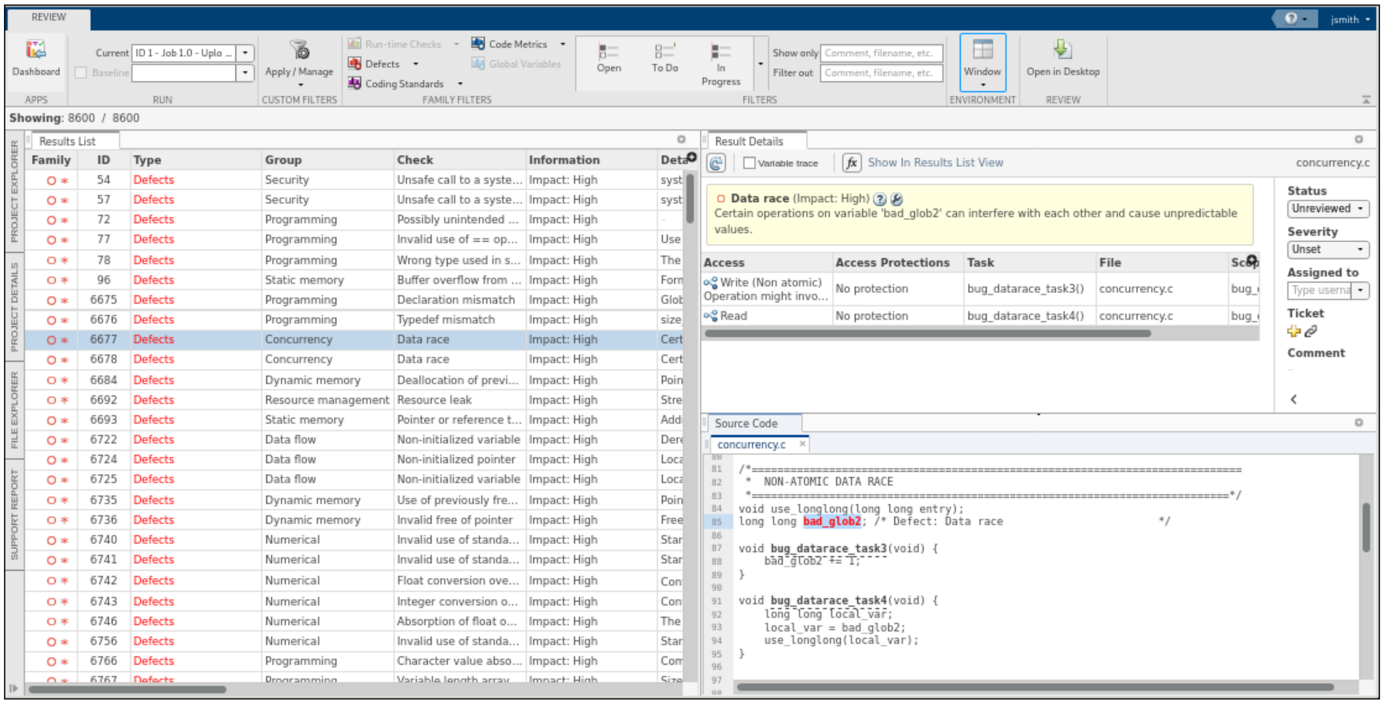
Assigned to
Type userna

Ticket

Comment

```
    local_var = good_glob1;
    END_CRITICAL_SECTION();
    (void)printf("%d", local_var);
}


/*=========================================================================
 *  NON-ATOMIC DATA RACE
 *=======================================================================*/
void use_longlong(long long entry);
long long bad_glob2; /* Defect: Data race                         */

void bug_datarace_task3(void) {
    bad_glob2 += 1;
}

void bug_datarace_task4(void) {
    long long local_var;
    local_var = bad_glob2;
    use_longlong(local_var);
}


long long good_glob2;

void corrected_datarace_task3(void) {
    BEGIN_CRITICAL_SECTION(); /* Fix: Protect with a critical section     */
    good_glob2 += 1;
    END_CRITICAL_SECTION();
}

void corrected_datarace_task4(void) {
    long long local_var;
    BEGIN_CRITICAL_SECTION();
    local_var = good_glob2;
    END_CRITICAL_SECTION();
    use_longlong(local_var);
}
```

The **Results Review** layout prioritizes the **Results List** and **Result Details** panes as you review and triage findings.



Current: ID 1 - Job 1.0 - Uplo ...

Show only: Comment, filename, etc.

Filter out: Comment, filename, etc.

Showing: 8600 / 8600

Family	ID	Type	Group	Check	Information	Deta
○ ✻	54	Defects	Security	Unsafe call to a syste...	Impact: High	syst
○ ✻	57	Defects	Security	Unsafe call to a syste...	Impact: High	syst
○ ✻	72	Defects	Programming	Possibly unintended ...	Impact: High	-
○ ✻	77	Defects	Programming	Invalid use of == op...	Impact: High	Use
○ ✻	78	Defects	Programming	Wrong type used in s...	Impact: High	The
○ ✻	96	Defects	Static memory	Buffer overflow from ...	Impact: High	Forr
○ ✻	6675	Defects	Programming	Declaration mismatch	Impact: High	Glol
○ ✻	6676	Defects	Programming	Typedef mismatch	Impact: High	size
○ ✻	6677	Defects	Concurrency	Data race	Impact: High	Cert
○ ✻	6678	Defects	Concurrency	Data race	Impact: High	Cert
○ ✻	6684	Defects	Dynamic memory	Deallocation of previ...	Impact: High	Poin
○ ✻	6692	Defects	Resource management	Resource leak	Impact: High	Stre
○ ✻	6693	Defects	Static memory	Pointer or reference t...	Impact: High	Add
○ ✻	6722	Defects	Data flow	Non-initialized variable	Impact: High	Dere
○ ✻	6724	Defects	Data flow	Non-initialized pointer	Impact: High	Loca
○ ✻	6725	Defects	Data flow	Non-initialized variable	Impact: High	Loca
○ ✻	6735	Defects	Dynamic memory	Use of previously fre...	Impact: High	Poin
○ ✻	6736	Defects	Dynamic memory	Invalid free of pointer	Impact: High	Free
○ ✻	6740	Defects	Numerical	Invalid use of standa...	Impact: High	Star
○ ✻	6741	Defects	Numerical	Invalid use of standa...	Impact: High	Star
○ ✻	6742	Defects	Numerical	Float conversion ove...	Impact: High	Con
○ ✻	6743	Defects	Numerical	Integer conversion o...	Impact: High	Con
○ ✻	6746	Defects	Numerical	Absorption of float o...	Impact: High	The
○ ✻	6756	Defects	Numerical	Invalid use of standa...	Impact: High	Star
○ ✻	6766	Defects	Programming	Character value abso...	Impact: High	Com
○ ✻	6767	Defects	Programming	Variable length array...	Impact: High	Size

concurrency.c

○ **Data race** (Impact: High)
Certain operations on variable 'bad_glob2' can interfere with each other and cause unpredictable values.

Access	Access Protections	Task	File	Sco
Write (Non atomic) Operation might invo...	No protection	bug_datarace_task3()	concurrency.c	bug_
Read	No protection	bug_datarace_task4()	concurrency.c	bug_

Status
Unreviewed

Severity
Unset

Assigned to
Type userna

Ticket

Comment

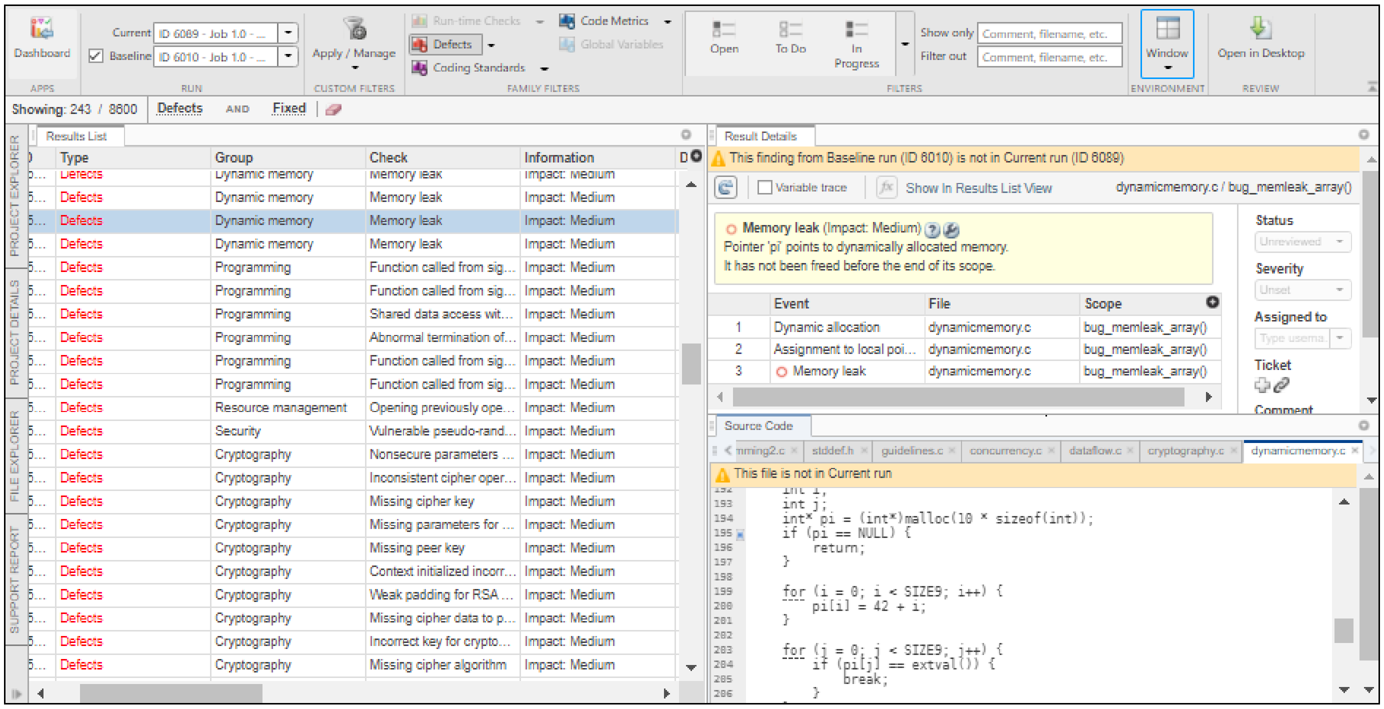
```
/*=========================================================================
 *  NON-ATOMIC DATA RACE
 *=======================================================================*/
void use_longlong(long long entry);
long long bad_glob2; /* Defect: Data race                         */

void bug_datarace_task3(void) {
    bad_glob2 += 1;
}

void bug_datarace_task4(void) {
    long long local_var;
    local_var = bad_glob2;
    use_longlong(local_var);
}
```

Code Quality Comparison Between Runs: Filter and view information for previous findings fixed in the current run

In R2021a, if you compare two project runs and some of the findings from the **Baseline** run are **Fixed** in the **Current** run, you can filter for and view the source code and result details for these findings. See “Compare Analysis Results to Previous Runs”.

Polyspace Access considers a finding **Fixed** if either:

- You make changes to your code that fix an issue.
- The source code that contains an issue is deleted or is not part of the current analysis.



Showing: 243 / 8800 Defects AND Fixed

Type	Group	Check	Information
Defects	Dynamic memory	Memory leak	Impact: Medium
Defects	Dynamic memory	Memory leak	Impact: Medium
Defects	Dynamic memory	Memory leak	Impact: Medium
Defects	Dynamic memory	Memory leak	Impact: Medium
Defects	Programming	Function called from sig...	Impact: Medium
Defects	Programming	Function called from sig...	Impact: Medium
Defects	Programming	Shared data access wit...	Impact: Medium
Defects	Programming	Abnormal termination of...	Impact: Medium
Defects	Programming	Function called from sig...	Impact: Medium
Defects	Programming	Function called from sig...	Impact: Medium
Defects	Resource management	Opening previously ope...	Impact: Medium
Defects	Security	Vulnerable pseudo-rand...	Impact: Medium
Defects	Cryptography	Nonsecure parameters ...	Impact: Medium
Defects	Cryptography	Inconsistent cipher oper...	Impact: Medium
Defects	Cryptography	Missing cipher key	Impact: Medium
Defects	Cryptography	Missing parameters for ...	Impact: Medium
Defects	Cryptography	Missing peer key	Impact: Medium
Defects	Cryptography	Context initialized incorr...	Impact: Medium
Defects	Cryptography	Weak padding for RSA ...	Impact: Medium
Defects	Cryptography	Missing cipher data to p...	Impact: Medium
Defects	Cryptography	Incorrect key for crypto...	Impact: Medium
Defects	Cryptography	Missing cipher algorithm	Impact: Medium

This finding from Baseline run (ID 6010) is not in Current run (ID 6089)

dynamicmemory.c / bug_memleak_array()

Memory leak (Impact: Medium)
Pointer 'pi' points to dynamically allocated memory.
It has not been freed before the end of its scope.

	Event	File	Scope
1	Dynamic allocation	dynamicmemory.c	bug_memleak_array()
2	Assignment to local poi...	dynamicmemory.c	bug_memleak_array()
3	Memory leak	dynamicmemory.c	bug_memleak_array()

This file is not in Current run

```
    int i;
    int j;
    int* pi = (int*)malloc(10 * sizeof(int));
    if (pi == NULL) {
        return;
    }

    for (i = 0; i < SIZE9; i++) {
        pi[i] = 42 + i;
    }

    for (j = 0; j < SIZE9; j++) {
        if (pi[j] == extval()) {
            break;
        }
```

Previously, you had to open the **Baseline** run as a separate tab to view the source code and result details for **Fixed** findings.

Installation

License Management: Uploading of results to Polyspace Access no longer requires a license checkout

In R2021a, the upload of analysis results to the Polyspace Access database does not trigger a Polyspace Access license checkout.

If you upload results as part of an automation script, you no longer consume a license when you run the script. Previously, each results upload triggered a license checkout.

User Manager: Enable pagination when requesting large set of users from LDAP server

In R2021a, if you use an LDAP server to retrieve user profiles and authenticate user logins, you can enable pagination to retrieve a large set of users from the LDAP server. See “Authenticate Users from Your Organization LDAP Server”.

Typically, LDAP servers limit the number of entries that they return in a result set. If the number of entries exceed that limit, the result set is truncated. When you enable pagination, the number of results is broken up into smaller sets. You are able to retrieve all entries from the LDAP server when you query a large set of users.

Bug Tracking Tool: Create Jira tickets for Jira projects that use single select custom fields

In R2021a, if you integrate the Jira software bug tracking tool (BTT) with Polyspace Access, you can create Jira tickets for Jira projects that are configured with single select custom fields. See “Configure Jira Software Bug Tracking Tool”.

Previously, Polyspace Access did not support the creation of Jira tickets in projects that used single select custom fields.

Admin Interface: Improved logging for Polyspace Access services

In R2021a, when you view the logs for the Polyspace Access services in the **Admin** user interface, the logs are automatically refreshed. You do not need to reload the page to view new events.

Polyspace As You Code

Bug Finder Analysis Engine for Single Source File: Run Polyspace as You Code analysis and view results in your IDE or code editor

In R2021a, you can use the new Polyspace as You Code capability to check your code for bugs and coding standard violations while you work in your IDE or code editor. The analysis runs on only the currently active file. You can identify and fix issues early in the development cycle.

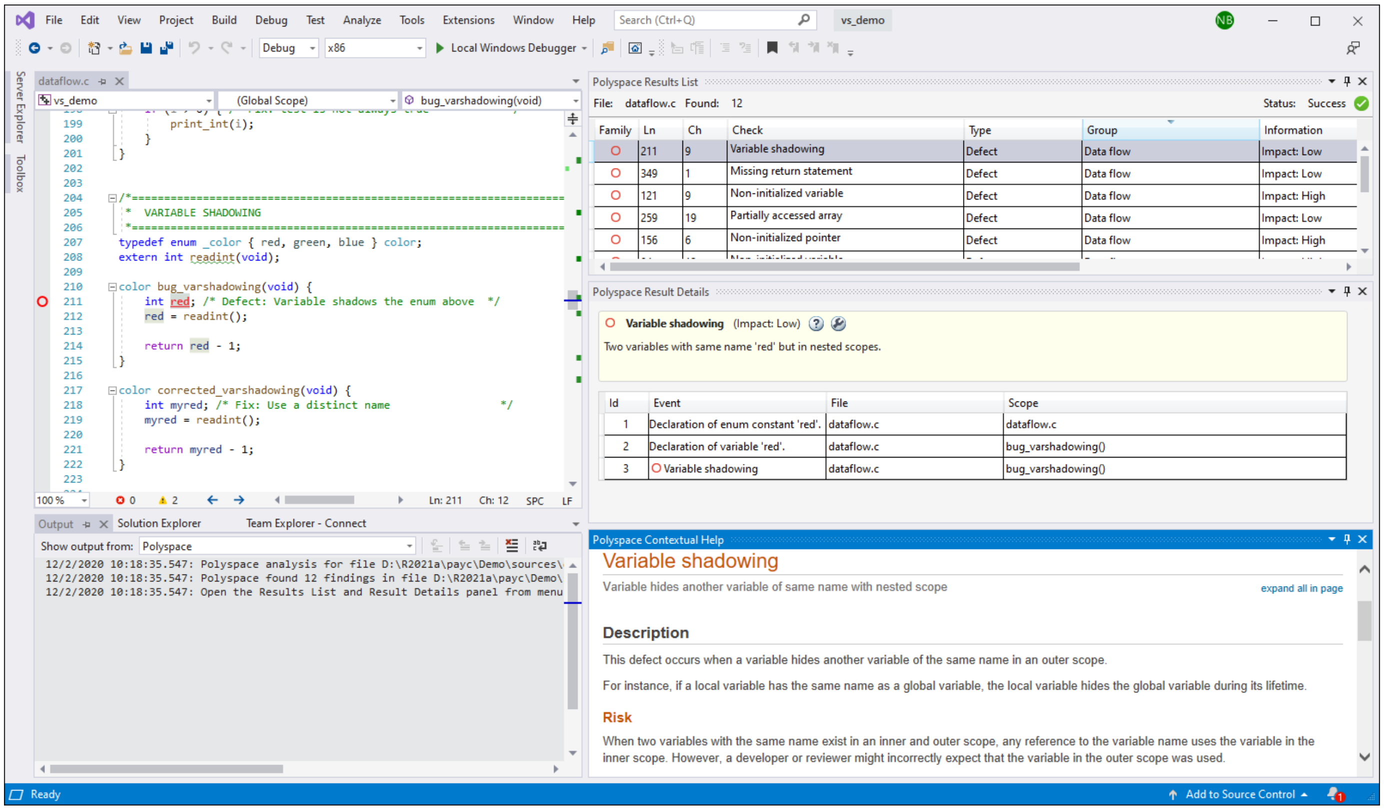
With Polyspace as You Code, you can:

- Start an analysis of the currently active file on save or on demand.
- Extract analysis options from your IDE project, your build command, or your JSON compilation database.
- Import analysis options from a Polyspace PSPRJ project file.
- Leverage results reviews from integration analyses uploaded in Polyspace Access to hide already justified results and focus on new findings.

The Polyspace as You Code analysis engine and IDE extensions are available for download from the Polyspace Access web interface. You must have a valid Polyspace Access license.

Polyspace Extension for Visual Studio: Run Polyspace As You Code analysis and view results in Visual Studio IDE

In R2021a, you can use the new Polyspace as You Code extension to check your code for bugs and coding standards violations while you code in your Visual Studio® IDE.



```
        print_int(i);
    }
}


/*==========================================================================
 *  VARIABLE SHADOWING
 *==========================================================================
typedef enum _color { red, green, blue } color;
extern int readint(void);

color bug_varshadowing(void) {
    int red; /* Defect: Variable shadows the enum above  */
    red = readint();

    return red - 1;
}

color corrected_varshadowing(void) {
    int myred; /* Fix: Use a distinct name              */
    myred = readint();

    return myred - 1;
}
```

Polyspace Results List

File: dataflow.c Found: 12 Status: Success

Family	Ln	Ch	Check	Type	Group	Information
○	211	9	Variable shadowing	Defect	Data flow	Impact: Low
○	349	1	Missing return statement	Defect	Data flow	Impact: Low
○	121	9	Non-initialized variable	Defect	Data flow	Impact: High
○	259	19	Partially accessed array	Defect	Data flow	Impact: Low
○	156	6	Non-initialized pointer	Defect	Data flow	Impact: High

Polyspace Result Details

○ **Variable shadowing** (Impact: Low)

Two variables with same name 'red' but in nested scopes.

Id	Event	File	Scope
1	Declaration of enum constant 'red'.	dataflow.c	dataflow.c
2	Declaration of variable 'red'.	dataflow.c	bug_varshadowing()
3	○ Variable shadowing	dataflow.c	bug_varshadowing()

Show output from: Polyspace

```
12/2/2020 10:18:35.547: Polyspace analysis for file D:\R2021a\payc\Demo\sources\
12/2/2020 10:18:35.547: Polyspace found 12 findings in file D:\R2021a\payc\Demo\
12/2/2020 10:18:35.547: Open the Results List and Result Details panel from menu
```

Polyspace Contextual Help

Variable shadowing

Variable hides another variable of same name with nested scope

expand all in page

Description

This defect occurs when a variable hides another variable of the same name in an outer scope.

For instance, if a local variable has the same name as a global variable, the local variable hides the global variable during its lifetime.

Risk

When two variables with the same name exist in an inner and outer scope, any reference to the variable name uses the variable in the inner scope. However, a developer or reviewer might incorrectly expect that the variable in the outer scope was used.

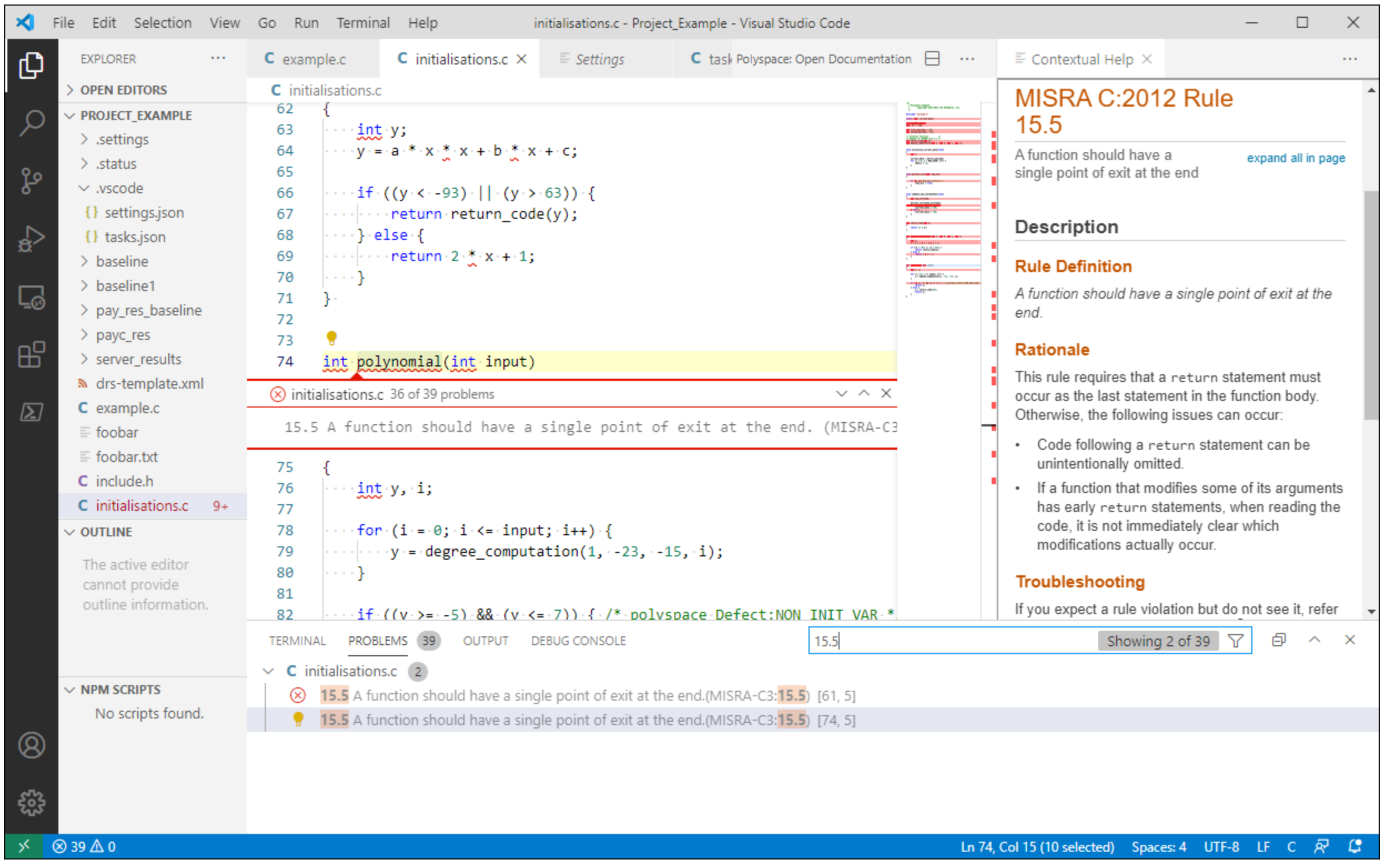
After you install the extension and the Polyspace as You Code analysis engine, you can:

- Start an analysis of the currently active file on save or on demand.
- Extract analysis options from your Visual Studio project or build command.
- Import analysis options from a Polyspace PSPRJ project file.
- View highlighted defects in your source code and apply annotations in one click.
- Sort results in the **Results List** and open the **Result Details** and **Contextual Help** to learn more about a defect.
- Leverage results reviews from integration analyses uploaded in Polyspace Access to hide already justified results and focus on new findings.

The Polyspace as You Code analysis engine and IDE extensions are available for download from the Polyspace Access web interface. You must have a valid Polyspace Access license.

Polyspace Extension for Visual Studio Code: Run Polyspace As You Code analysis and view results in Visual Studio Code code editor

In R2021a, you can use the new Polyspace as You Code extension to check your code for bugs and coding standards violations while you code in your Visual Studio Code editor.



```
{
    int y;
    y = a * x * x + b * x + c;

    if ((y < -93) || (y > 63)) {
        return return_code(y);
    } else {
        return 2 * x + 1;
    }
}

int polynomial(int input)
```

initialisations.c 36 of 39 problems

```
15.5 A function should have a single point of exit at the end. (MISRA-C3
```

```
{
    int y, i;

    for (i = 0; i <= input; i++) {
        y = degree_computation(1, -23, -15, i);
    }

    if ((y >= -5) && (y <= 7)) { /* polyspace Defect:NON_INIT_VAR *
```

initialisations.c 2

15.5 A function should have a single point of exit at the end.(MISRA-C3:15.5) [61, 5]

15.5 A function should have a single point of exit at the end.(MISRA-C3:15.5) [74, 5]

MISRA C:2012 Rule 15.5

A function should have a single point of exit at the end

expand all in page

Description

Rule Definition

A function should have a single point of exit at the end.

Rationale

This rule requires that a `return` statement must occur as the last statement in the function body. Otherwise, the following issues can occur:

- Code following a `return` statement can be unintentionally omitted.
- If a function that modifies some of its arguments has early `return` statements, when reading the code, it is not immediately clear which modifications actually occur.

Troubleshooting

If you expect a rule violation but do not see it, refer

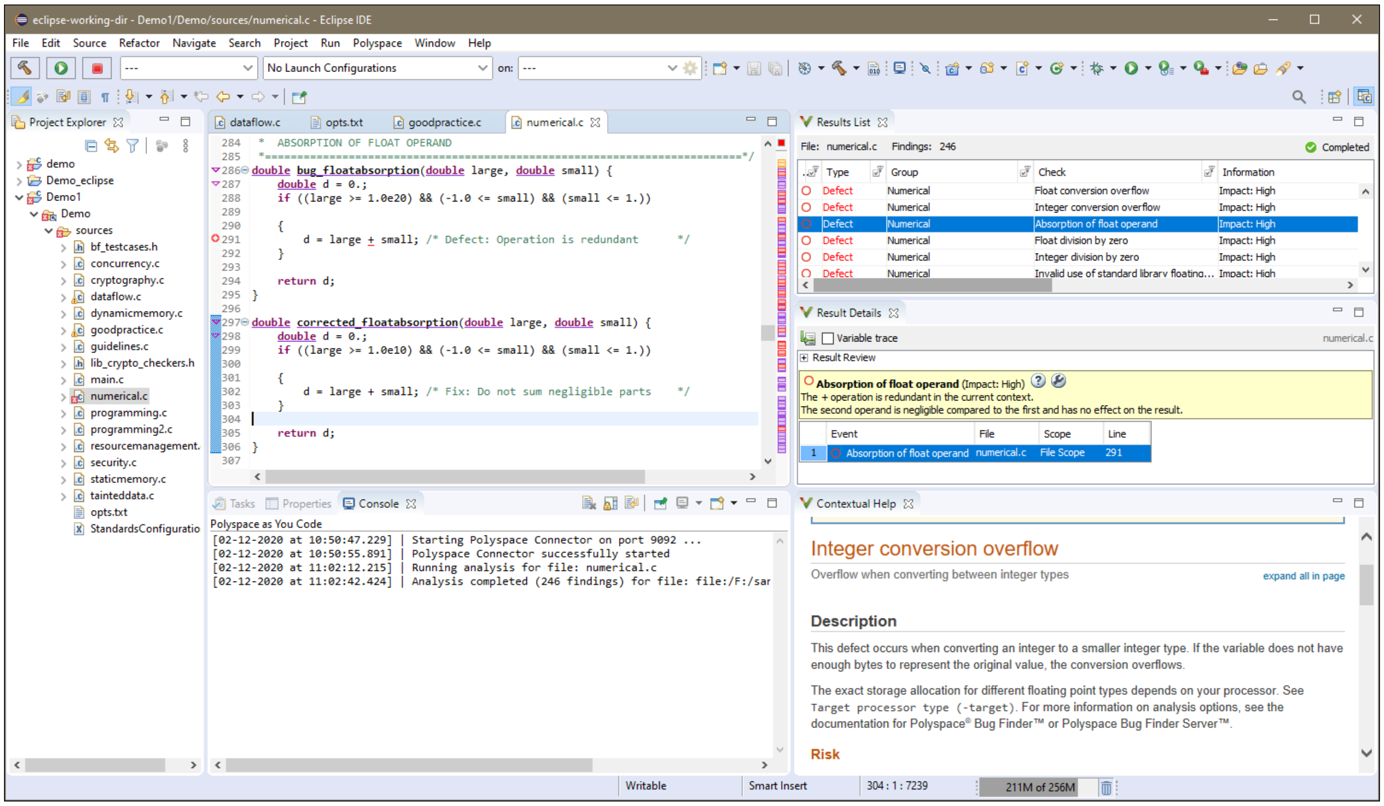
After you install the extension and the Polyspace as You Code, you can:

- Start an analysis of the currently active file on save or on demand.
- Extract analysis options from your Visual Studio Code build task or build command.
- Import analysis options from a Polyspace PSPRJ project file.
- View highlighted defects in your source code and apply annotations in one click.
- Filter results in the **Problems** pane and open the **Contextual Help** to learn more about a defect.
- Leverage results reviews from integration analyses uploaded in Polyspace Access to hide already justified results and focus on new findings.

The Polyspace as You Code analysis engine and IDE extensions are available for download from the Polyspace Access web interface. You must have a valid Polyspace Access license.

Polyspace Extension for Eclipse: Run Polyspace As You Code analysis and view results in Eclipse IDE

In R2021a, you can use the new Polyspace as You Code extension to check your code for bugs and coding standards violations while you code in your Eclipse™ IDE.



```
 *  ABSORPTION OF FLOAT OPERAND
 *==========================================================================*/
double bug_floatabsorption(double large, double small) {
    double d = 0.;
    if ((large >= 1.0e20) && (-1.0 <= small) && (small <= 1.))

    {
        d = large + small; /* Defect: Operation is redundant      */
    }

    return d;
}

double corrected_floatabsorption(double large, double small) {
    double d = 0.;
    if ((large >= 1.0e10) && (-1.0 <= small) && (small <= 1.))

    {
        d = large + small; /* Fix: Do not sum negligible parts    */
    }

    return d;
}
```

Polyspace as You Code

```
[02-12-2020 at 10:50:47.229] | Starting Polyspace Connector on port 9092 ...
[02-12-2020 at 10:50:55.891] | Polyspace Connector successfully started
[02-12-2020 at 11:02:12.215] | Running analysis for file: numerical.c
[02-12-2020 at 11:02:42.424] | Analysis completed (246 findings) for file: file:/F:/sar
```

File: numerical.c Findings: 246 Completed

Type	Group	Check	Information
Defect	Numerical	Float conversion overflow	Impact: High
Defect	Numerical	Integer conversion overflow	Impact: High
Defect	Numerical	Absorption of float operand	Impact: High
Defect	Numerical	Float division by zero	Impact: High
Defect	Numerical	Integer division by zero	Impact: High
Defect	Numerical	Invalid use of standard library floating...	Impact: High

Variable trace numerical.c

Result Review

Absorption of float operand (Impact: High)
The + operation is redundant in the current context.
The second operand is negligible compared to the first and has no effect on the result.

	Event	File	Scope	Line
1	Absorption of float operand	numerical.c	File Scope	291

Integer conversion overflow

Overflow when converting between integer types

expand all in page

Description

This defect occurs when converting an integer to a smaller integer type. If the variable does not have enough bytes to represent the original value, the conversion overflows.

The exact storage allocation for different floating point types depends on your processor. See `Target processor type (-target)`. For more information on analysis options, see the documentation for Polyspace® Bug Finder™ or Polyspace Bug Finder Server™.

Risk

After you install the extension, you can:

- Start an analysis of the currently active file on save or on demand.
- Extract analysis options from your Eclipse project or build command.
- Import analysis options from a Polyspace PSPRJ project file.
- View highlighted defects in your source code and apply annotations in one click.
- Sort results in the **Results List** and open the **Result Details** and **Contextual Help** to learn more about a defect.
- Leverage results reviews from integration analyses uploaded in Polyspace Access to hide already justified results and focus on new findings.

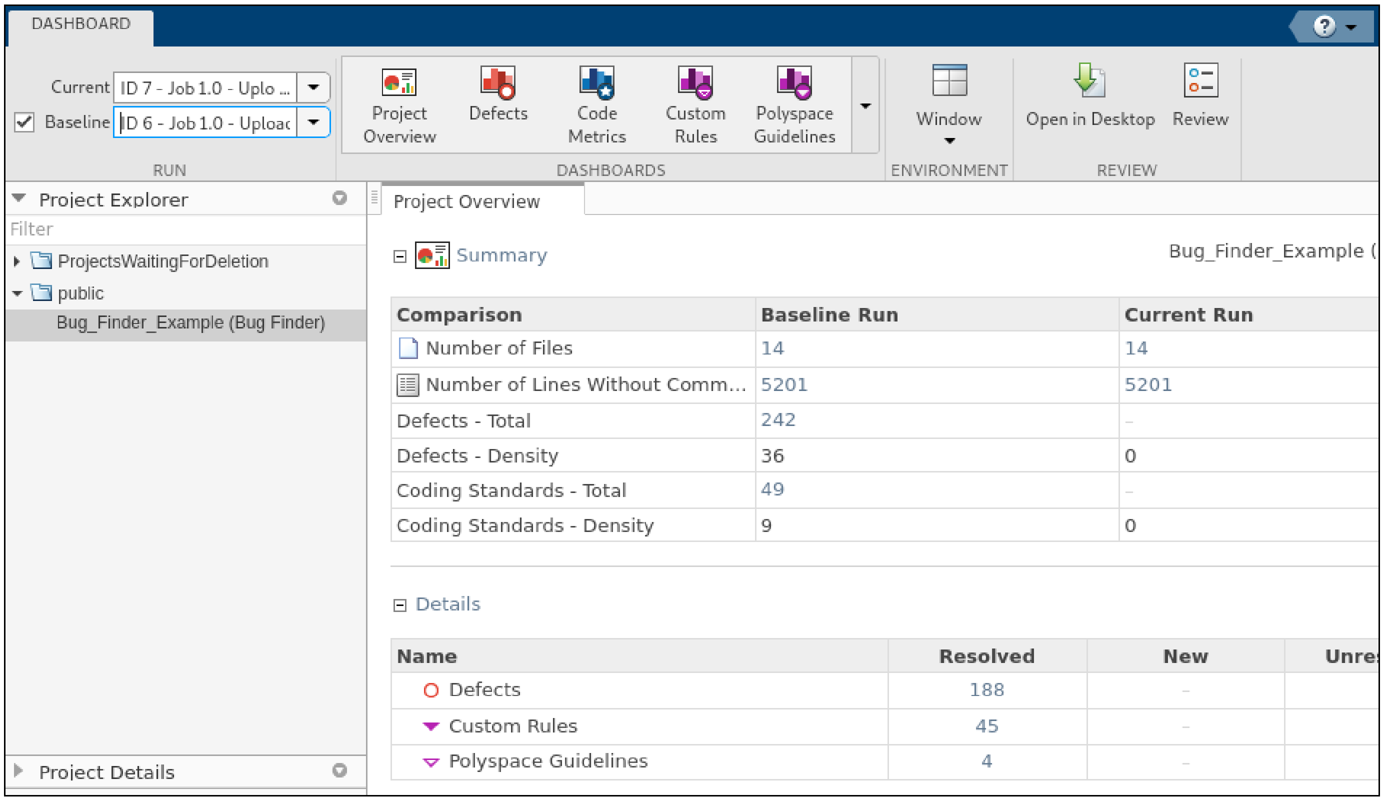
The Polyspace as You Code analysis engine and IDE extensions are available for download from the Polyspace Access web interface. You must have a valid Polyspace Access license.

R2020b

Dashboard and Review in Web Browser

Code Quality Improvement Progress: Compare results from current run to previous runs and determine progress in code quality improvement

In R2020b, you can select any two runs of a project in the Polyspace web interface (current and baseline runs) and compare them. You can compare a current run to only older baseline runs.



Comparison	Baseline Run	Current Run
Number of Files	14	14
Number of Lines Without Comm...	5201	5201
Defects - Total	242	-
Defects - Density	36	0
Coding Standards - Total	49	-
Coding Standards - Density	9	0

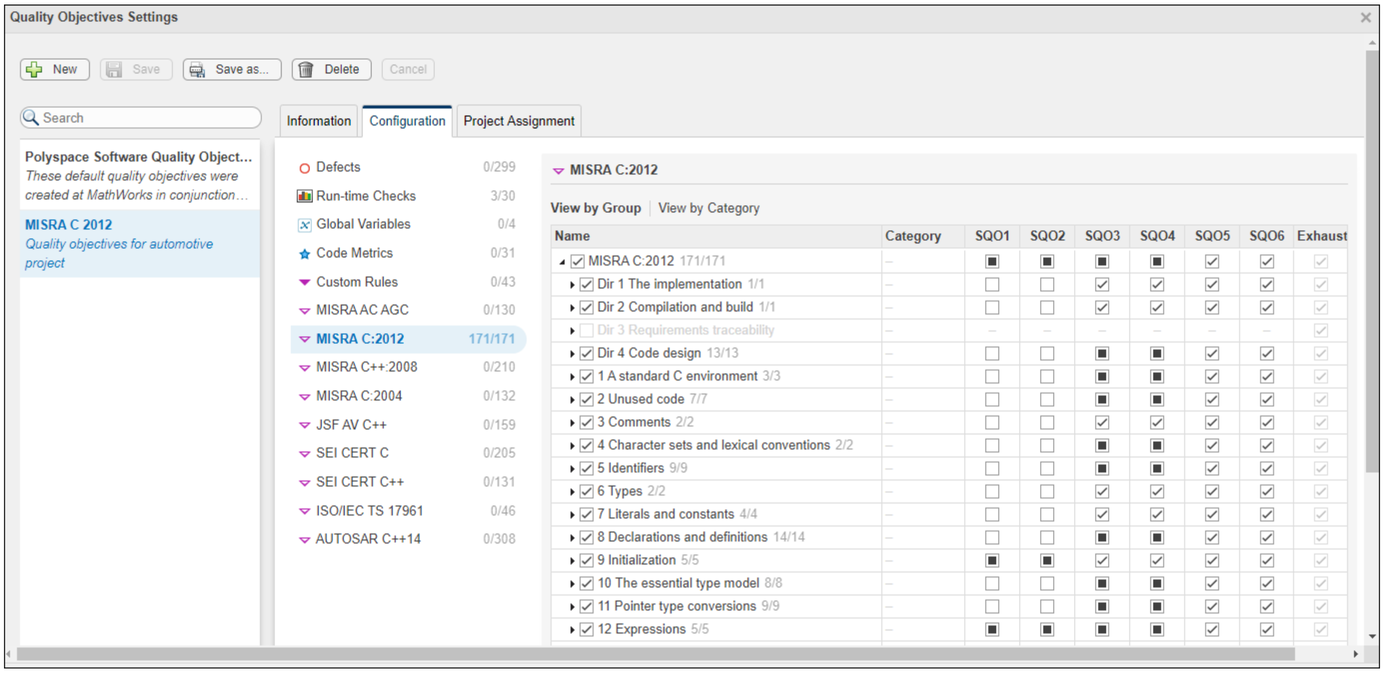
Name	Resolved	New	Unre
Defects	188	-	
Custom Rules	45	-	
Polyspace Guidelines	4	-	

The comparison shows the number of analysis findings that are:

- **Resolved**. Findings from the baseline run no longer found in the current run.
- **New**. Findings in the current run that were not present in the baseline run.
- **Unresolved**. Findings from the baseline run that are still present in the current run.

Code Quality Objectives: Define custom quality objectives definitions and apply them to specific projects

In R2020b, you can create custom quality objectives definitions and apply those definitions to specific projects. For instance, if you want to track the compliance of a project with a coding standard, you can create Quality Objective thresholds for that coding standard and apply them to your project.



Quality Objectives Settings

New | Save | Save as... | Delete | Cancel

Search

Polyspace Software Quality Object...
These default quality objectives were created at MathWorks in conjunction...

MISRA C 2012
Quality objectives for automotive project

Information | Configuration | Project Assignment

Defects	0/299
Run-time Checks	3/30
Global Variables	0/4
Code Metrics	0/31
Custom Rules	0/43
MISRA AC AGC	0/130
MISRA C:2012	171/171
MISRA C++:2008	0/210
MISRA C:2004	0/132
JSF AV C++	0/159
SEI CERT C	0/205
SEI CERT C++	0/131
ISO/IEC TS 17961	0/46
AUTOSAR C++14	0/308

MISRA C:2012

View by Group | View by Category

Name	Category	SQO1	SQO2	SQO3	SQO4	SQO5	SQO6	Exhaust
✓ MISRA C:2012 171/171	–	■	■	■	■	✓	✓	✓
✓ Dir 1 The implementation 1/1	–	☐	☐	✓	✓	✓	✓	✓
✓ Dir 2 Compilation and build 1/1	–	☐	☐	✓	✓	✓	✓	✓
☐ Dir 3 Requirements traceability	–	–	–	–	–	–	–	✓
✓ Dir 4 Code design 13/13	–	☐	☐	■	■	✓	✓	✓
✓ 1 A standard C environment 3/3	–	☐	☐	■	■	✓	✓	✓
✓ 2 Unused code 7/7	–	☐	☐	■	■	✓	✓	✓
✓ 3 Comments 2/2	–	☐	☐	✓	✓	✓	✓	✓
✓ 4 Character sets and lexical conventions 2/2	–	☐	☐	■	■	✓	✓	✓
✓ 5 Identifiers 9/9	–	☐	☐	■	■	✓	✓	✓
✓ 6 Types 2/2	–	☐	☐	✓	✓	✓	✓	✓
✓ 7 Literals and constants 4/4	–	☐	☐	✓	✓	✓	✓	✓
✓ 8 Declarations and definitions 14/14	–	☐	☐	■	■	✓	✓	✓
✓ 9 Initialization 5/5	–	■	■	✓	✓	✓	✓	✓
✓ 10 The essential type model 8/8	–	☐	☐	■	■	✓	✓	✓
✓ 11 Pointer type conversions 9/9	–	☐	☐	■	■	✓	✓	✓
✓ 12 Expressions 5/5	–	■	■	■	■	✓	✓	✓

To create custom quality objectives definitions, you must be an **Administrator** or **Owner**.

Previously, custom quality objectives applied to all projects.

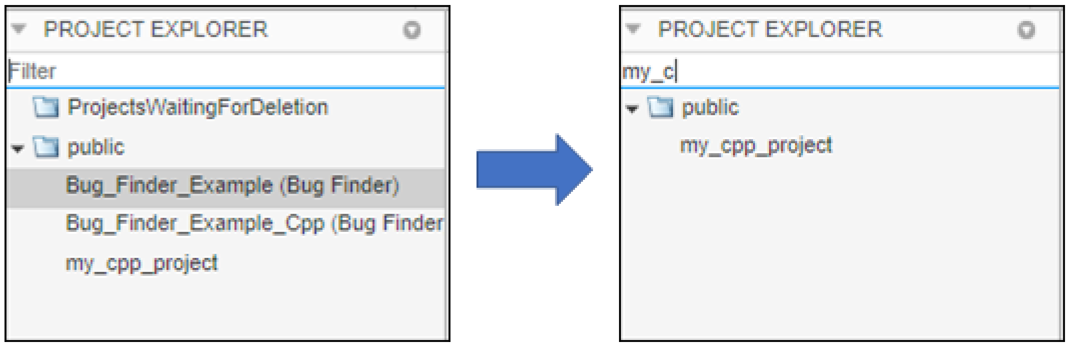
Source Code Tooltips: Display only information necessary to understand the selected defect

In R2020b, Bug Finder tooltips show only information that is necessary to understand the currently selected defect.

Previously, tooltips showed range information, such as all possible values of a specific variable in the given context. You can still see this range information in Code Prover.

Project Selection: Find a project in the Project Explorer through a text filter

In R2020b, you can use a text filter in the **Project Explorer** to find projects that are not visible in a folder hierarchy. The text filter is not case sensitive.


PROJECT EXPLORER
Filter
ProjectsWaitingForDeletion
public
Bug_Finder_Example (Bug Finder)
Bug_Finder_Example_Cpp (Bug Finder
my_cpp_project
PROJECT EXPLORER
my_c
public
my_cpp_project

Installation

Bug Tracking Tool: Integrate with Jira Software Cloud

In R2020b, you can integrate Jira Software Cloud with Polyspace Access. After you configure Polyspace Access, 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. See Configure Issue Tracker (Polyspace Bug Finder Access) (Polyspace Code Prover Access).

Previously, you could integrate Polyspace Access with only self-managed Jira Software.

Cluster Admin Settings: Validate values of settings on demand or on save

In R2020b, the **Cluster Admin** validates the settings that you enter in the **Cluster Settings** when you save those settings. You can also validate the settings before you save by clicking **Validate now** at the bottom of the page.

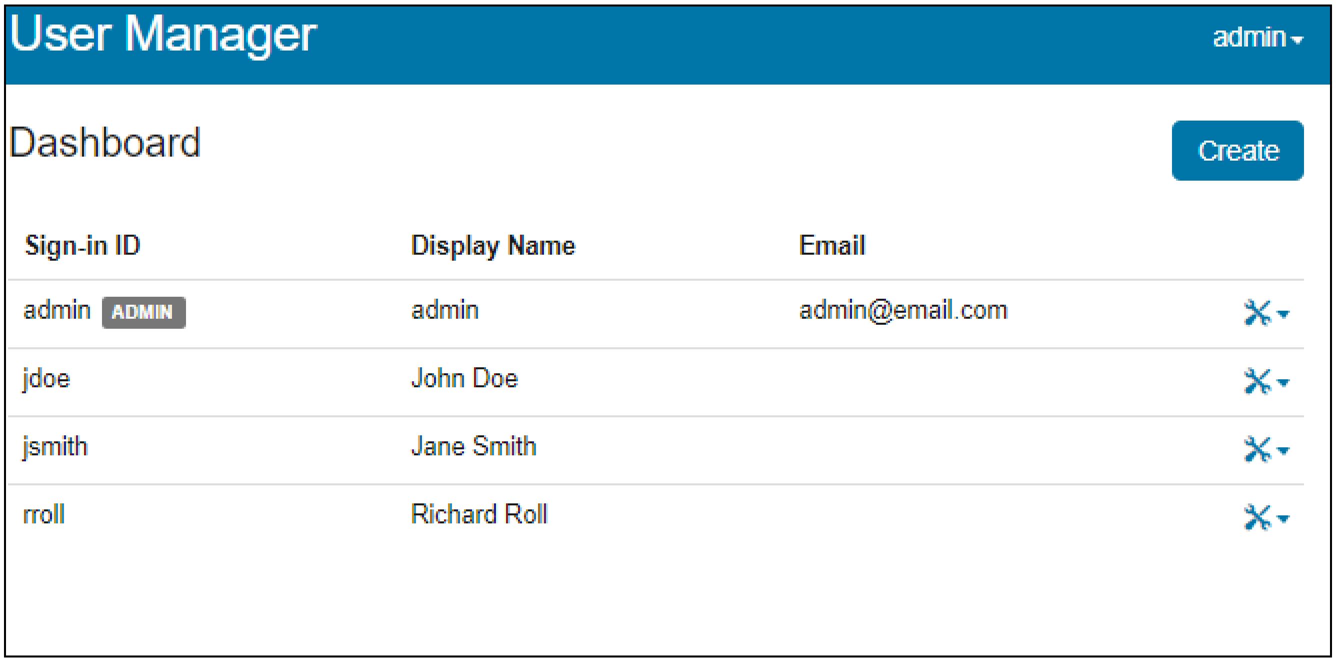
HTTPS Configuration: Configure services without specifying ports or SSL certificates

In R2020b, if you install Polyspace Access on a single node, the ports of the Polyspace Access services are no longer exposed. You do not need to specify port numbers for the services or to provide SSL private keys and certificates for the HTTPS configuration. See Configure Polyspace Access for HTTPS (Polyspace Bug Finder Access) (Polyspace Code Prover Access)

Previously, you had to check the availability of the ports for the services, and then you provided a private key and SSL certificate file to enable the HTTPS protocol for Polyspace Access.

Functionality Replaced: Polyspace Access embedded LDAP

The Polyspace Access embedded LDAP is removed in R2020b. To continue using custom login credentials for Polyspace Access, use the **User Manager** internal directory instead. See (Polyspace Code Prover Access)Authenticate Users from Internal Directory (Polyspace Bug Finder Access).



Dashboard

Sign-in ID	Display Name	Email
admin ADMIN	admin	admin@email.com
jdoe	John Doe	
jsmith	Jane Smith	
rroll	Richard Roll	

Compatibility Considerations

In the **User Manager** interface, create users to transfer the user names and passwords that you stored in the embedded LDAP LDIF file to the **User Manager** database.

Changes in Polyspace Access docker containers, options, and binaries

In R2020b, the following docker containers, options, and binaries have been renamed:

- The `cop-docker-agent` binary is now called the `admin-docker-agent`
- **HTTPS Options**

Previous Option Name	Current Option Name
`--https-certificate-file`	`--ssl-cert-file`
`--https-private-key-file`	`--ssl-key-file`
`--https-trusted-certificates-file`	`--ssl-ca-file`

- **Containers**

Previous Container Name	Current Container Name
`polyspace-db`	`polyspace-access-db-main`
`polyspace-etl`	`polyspace-access-etl-main`
`polyspace-gateway`	`gateway`
`polyspace-issuetracker`	`issuetracker-server-main`
`polyspace-web-server`	`polyspace-access-web-server-main`

Compatibility Considerations

In your scripts, replace instances of the previous names with the current names. You cannot reuse a settings configuration file (`settings.json`) from a previous release of Polyspace Access with the R2020b software.

R2020a

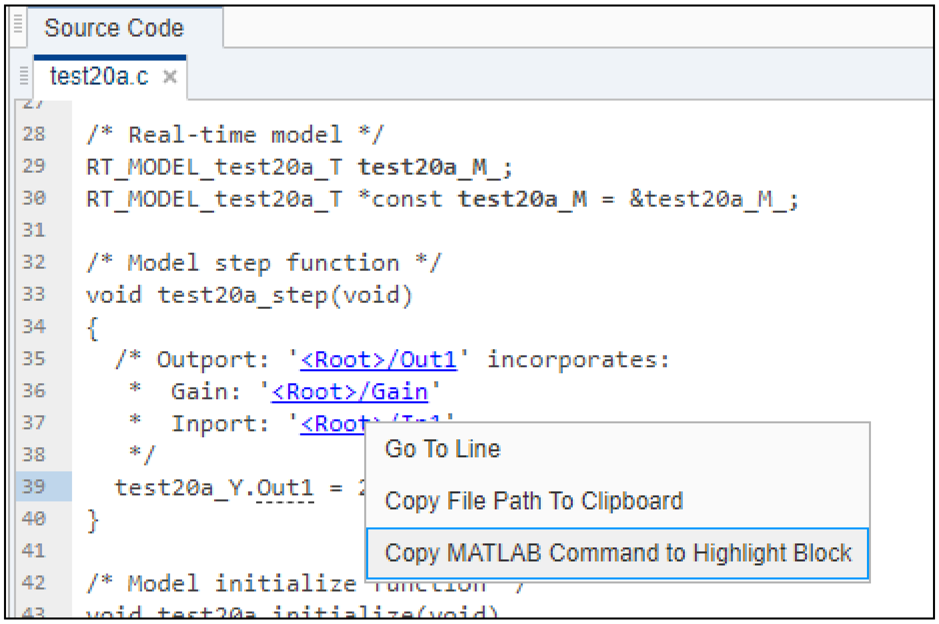
Dashboard and Review in Web Browser

Simulink Support: Navigate from generated code in Polyspace Access to blocks in model

In R2020a, if you run Polyspace on generated code in Simulink® and upload the results to Polyspace Access, you can navigate from the source code in Polyspace Access to blocks in the model.

On the **Source Code** pane in the Polyspace Access web interface, links in code comments show blocks that generate the subsequent lines of code. To see the block in the model:

1 Right-click a link and select **Copy MATLAB Command to Highlight Block**.


Source Code
test20a.c
Go To Line
Copy File Path To Clipboard
Copy MATLAB Command to Highlight Block

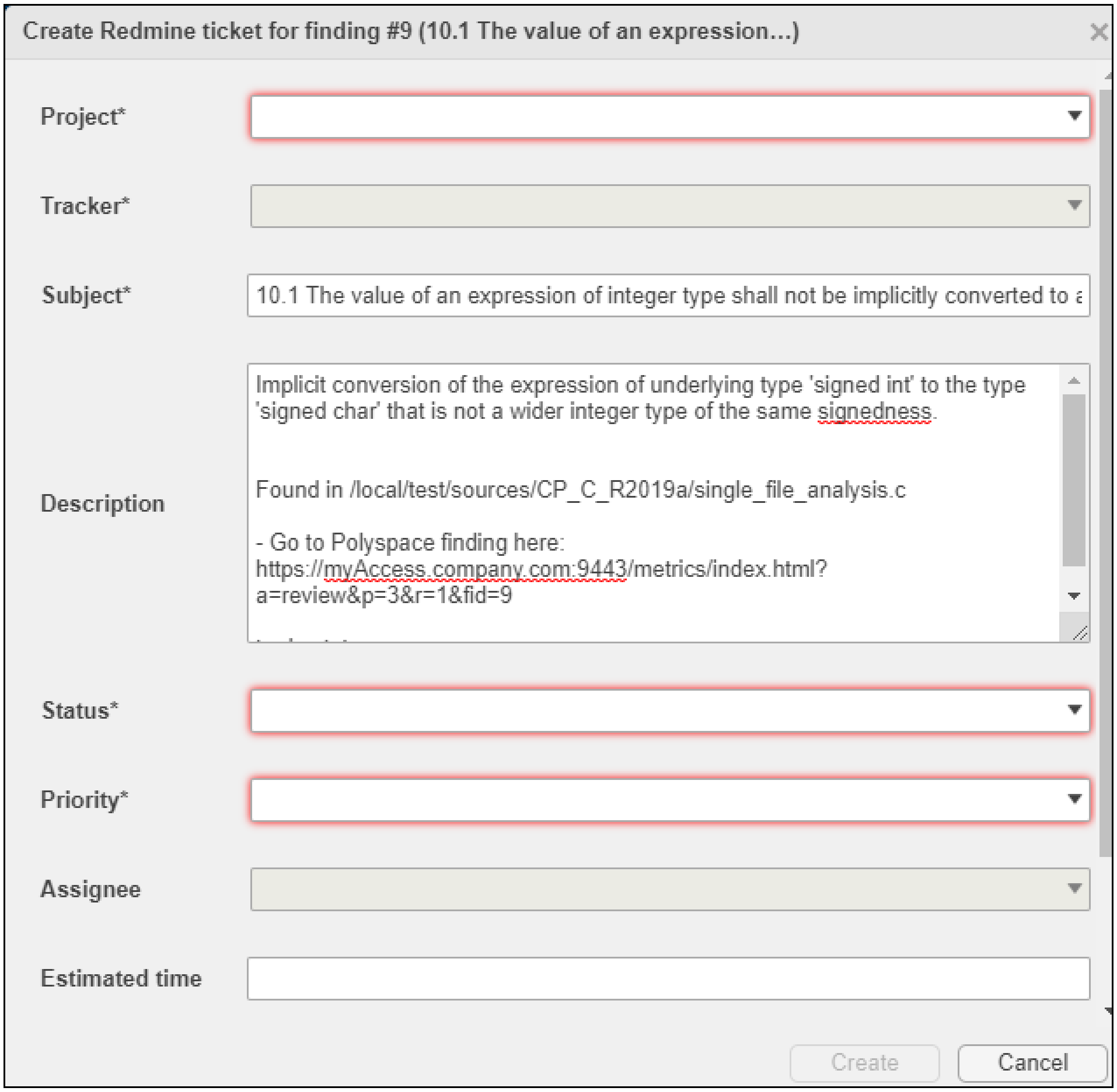
```
/* Real-time model */
RT_MODEL_test20a_T test20a_M_;
RT_MODEL_test20a_T *const test20a_M = &test20a_M_;

/* Model step function */
void test20a_step(void)
{
  /* Outport: '<Root>/Out1' incorporates:
   *  Gain: '<Root>/Gain'
   *  Inport: '<Root
   */
  test20a_Y.Out1 = 2
}

/* Model initialize
void test20a_initialize(void)
```

This action copies the MATLAB® command required to highlight the block. The command uses the `Simulink.ID.hilite` function.

2 In MATLAB, with the model open, paste and run the copied command.

Bug Tracking Tool Support: Create Redmine tickets for Polyspace Access results and assign to developers

In R2020a, Polyspace Access supports integration with the Redmine bug tracking tool. If you use Redmine, after you configure Polyspace Access, you can create a Redmine 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 Redmine project.


Create Redmine ticket for finding #9 (10.1 The value of an expression...)

Project*	
Tracker*	
Subject*	10.1 The value of an expression of integer type shall not be implicitly converted to a
Description	Implicit conversion of the expression of underlying type 'signed int' to the type 'signed char' that is not a wider integer type of the same signedness. Found in /local/test/sources/CP_C_R2019a/single_file_analysis.c - Go to Polyspace finding here: https://myAccess.company.com:9443/metrics/index.html?a=review&p=3&r=1&fid=9
Status*	
Priority*	
Assignee	
Estimated time	

Once you create a ticket, the **Result Details** pane displays a link that you can click to open the ticket in the Redmine interface. See also Track Issue in Bug Tracking Tool.

Bug Tracking Tool Support: Manage tickets for multiple findings

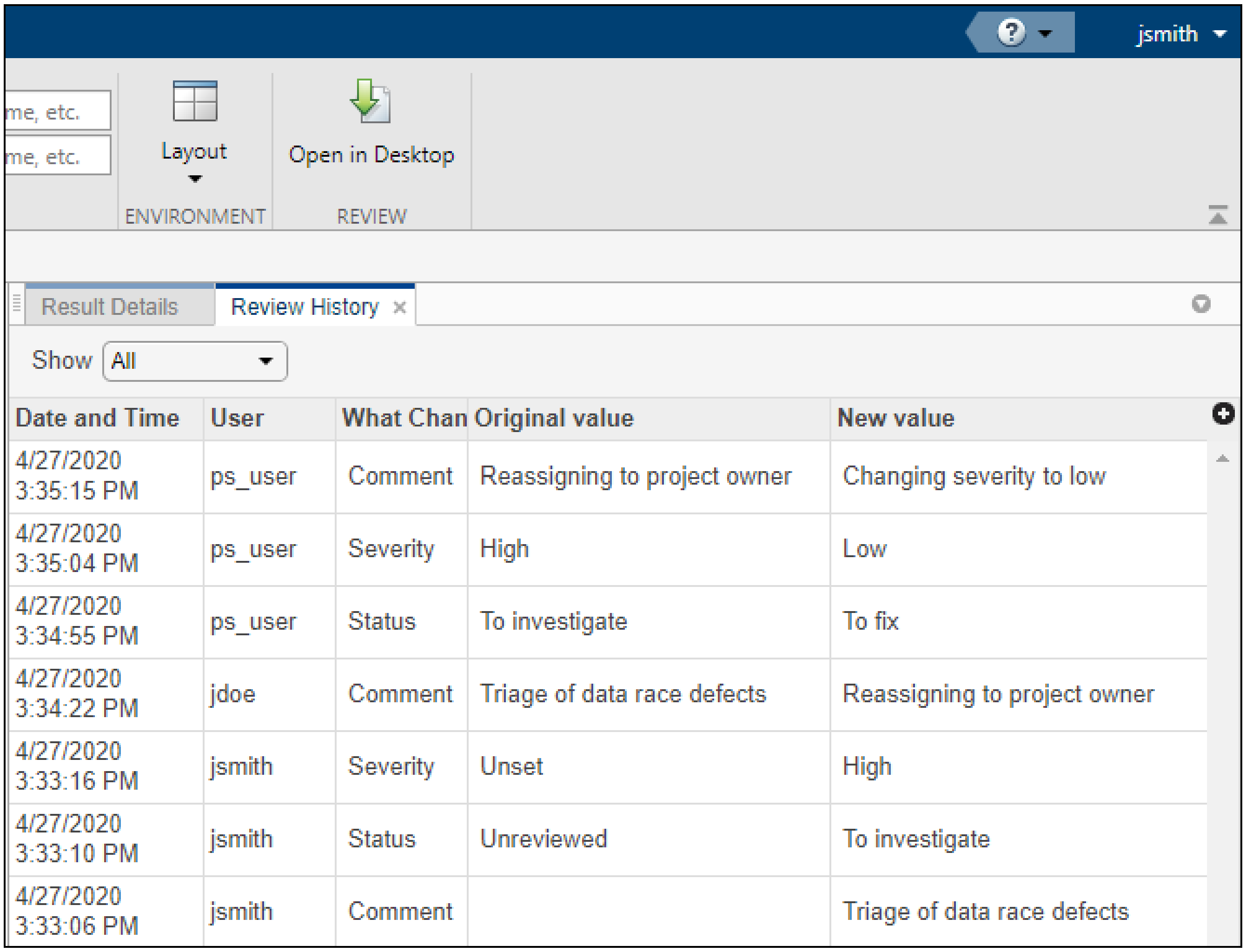
In R2020a, if you create a bug tracking tool ticket in Polyspace Access, you can select multiple findings that you associate with the ticket. If a ticket already exists, you can add that ticket to additional findings or you can detach the ticket from findings that are associated with the ticket.

Previously, you could create a ticket for only one finding at a time and you could not detach a ticket from a finding.

For more information, see Track Issue in Bug Tracking Tool.

Results Review: See review history of findings

In R2020a, you can open the **Review History** pane to see all the changes to the review fields of findings with a timestamp and the name of the user who made the change. On the Polyspace Access toolstrip, select **Layout > Show/Hide View**.



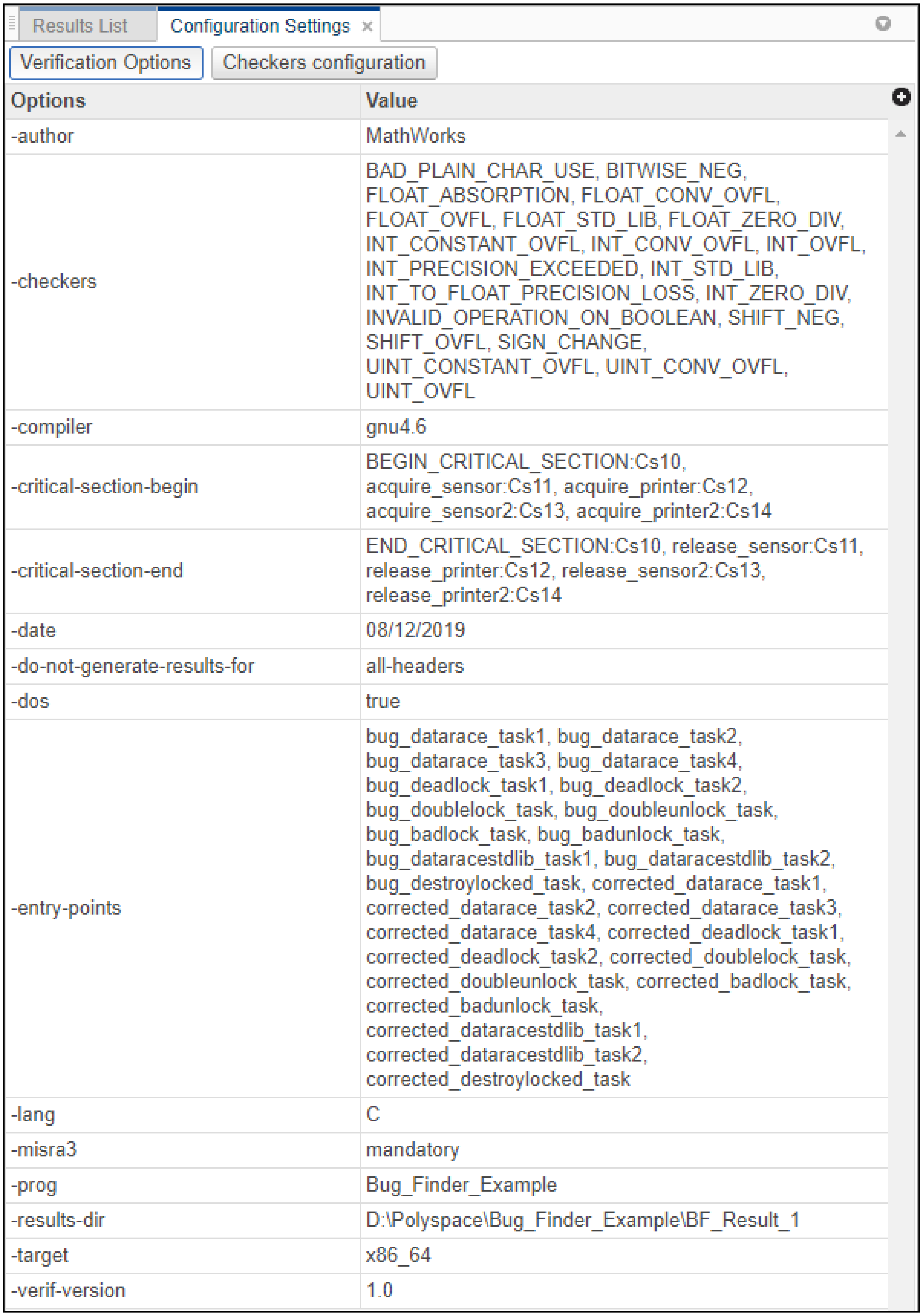
Date and Time	User	What Chan	Original value	New value
4/27/2020 3:35:15 PM	ps_user	Comment	Reassigning to project owner	Changing severity to low
4/27/2020 3:35:04 PM	ps_user	Severity	High	Low
4/27/2020 3:34:55 PM	ps_user	Status	To investigate	To fix
4/27/2020 3:34:22 PM	jdoe	Comment	Triage of data race defects	Reassigning to project owner
4/27/2020 3:33:16 PM	jsmith	Severity	Unset	High
4/27/2020 3:33:10 PM	jsmith	Status	Unreviewed	To investigate
4/27/2020 3:33:06 PM	jsmith	Comment		Triage of data race defects

You can use this information to better understand how and why the **Severity** or **Status** of a finding has changed, and retrieve previous comments that were overwritten.

For more information, see Review History.

Results Review: See the configuration options used for analysis

In R2020a, you can open the **Configuration Settings** pane to view the Polyspace configuration options that were enabled to generate the analysis results. On the Polyspace Access toolstrip, select **Layout > Show/Hide View**.



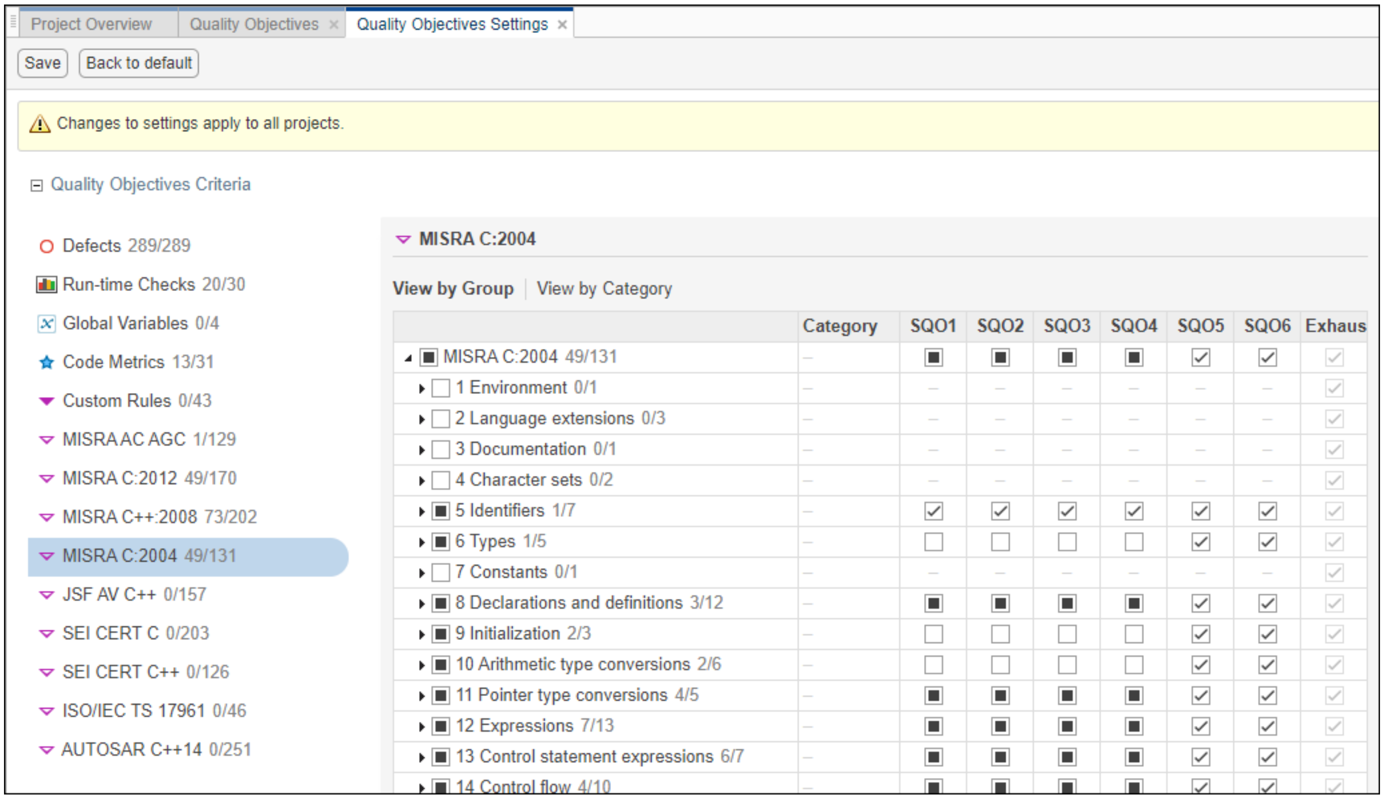
Options	Value
-author	MathWorks
-checkers	BAD_PLAIN_CHAR_USE, BITWISE_NEG, FLOAT_ABSORPTION, FLOAT_CONV_OVFL, FLOAT_OVFL, FLOAT_STD_LIB, FLOAT_ZERO_DIV, INT_CONSTANT_OVFL, INT_CONV_OVFL, INT_OVFL, INT_PRECISION_EXCEEDED, INT_STD_LIB, INT_TO_FLOAT_PRECISION_LOSS, INT_ZERO_DIV, INVALID_OPERATION_ON_BOOLEAN, SHIFT_NEG, SHIFT_OVFL, SIGN_CHANGE, UINT_CONSTANT_OVFL, UINT_CONV_OVFL, UINT_OVFL
-compiler	gnu4.6
-critical-section-begin	BEGIN_CRITICAL_SECTION:Cs10, acquire_sensor:Cs11, acquire_printer:Cs12, acquire_sensor2:Cs13, acquire_printer2:Cs14
-critical-section-end	END_CRITICAL_SECTION:Cs10, release_sensor:Cs11, release_printer:Cs12, release_sensor2:Cs13, release_printer2:Cs14
-date	08/12/2019
-do-not-generate-results-for	all-headers
-dos	true
-entry-points	bug_datarace_task1, bug_datarace_task2, bug_datarace_task3, bug_datarace_task4, bug_deadlock_task1, bug_deadlock_task2, bug_doublelock_task, bug_doubleunlock_task, bug_badlock_task, bug_badunlock_task, bug_dataracestdlib_task1, bug_dataracestdlib_task2, bug_destroylocked_task, corrected_datarace_task1, corrected_datarace_task2, corrected_datarace_task3, corrected_datarace_task4, corrected_deadlock_task1, corrected_deadlock_task2, corrected_doublelock_task, corrected_doubleunlock_task, corrected_badlock_task, corrected_badunlock_task, corrected_dataracestdlib_task1, corrected_dataracestdlib_task2, corrected_destroylocked_task
-lang	C
-misra3	mandatory
-prog	Bug_Finder_Example
-results-dir	D:\Polyspace\Bug_Finder_Example\BF_Result_1
-target	x86_64
-verif-version	1.0

You can use this information to better understand your results. For instance, you might expect to see a certain coding rule violation but the checker for this rule is not enabled. Previously, you had to parse the **Run Log** to see which options and checkers were enabled.

For more information, see Configuration Settings.

Code Quality Objectives: Customize thresholds used to track the quality of your code

In R2020a, if you use Quality Objectives to track the quality of your code, you can customize the thresholds you use as pass/fail criteria to better align with your company or project requirements. For instance, you can define quality gates to ensure adherence to a specific external coding standard.



Changes to settings apply to all projects.

Quality Objectives Criteria

- Defects 289/289
- Run-time Checks 20/30
- Global Variables 0/4
- Code Metrics 13/31
- Custom Rules 0/43
- MISRA AC AGC 1/129
- MISRA C:2012 49/170
- MISRA C++:2008 73/202
- MISRA C:2004 49/131
- JSF AV C++ 0/157
- SEI CERT C 0/203
- SEI CERT C++ 0/126
- ISO/IEC TS 17961 0/46
- AUTOSAR C++14 0/251

MISRA C:2004

View by Group | View by Category

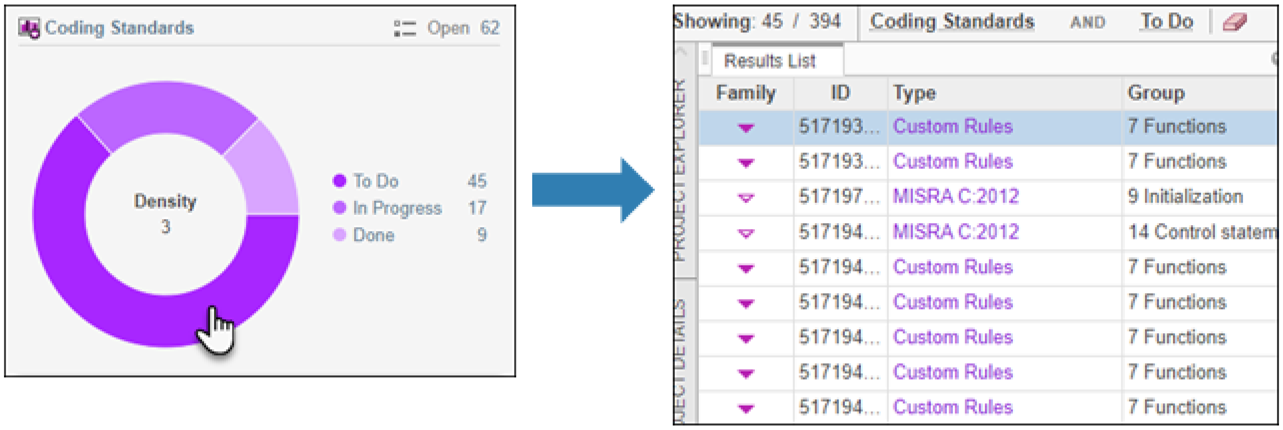
	Category	SQO1	SQO2	SQO3	SQO4	SQO5	SQO6	Exhaus
■ MISRA C:2004 49/131	–	■	■	■	■	✓	✓	✓
☐ 1 Environment 0/1	–	–	–	–	–	–	–	✓
☐ 2 Language extensions 0/3	–	–	–	–	–	–	–	✓
☐ 3 Documentation 0/1	–	–	–	–	–	–	–	✓
☐ 4 Character sets 0/2	–	–	–	–	–	–	–	✓
■ 5 Identifiers 1/7	–	✓	✓	✓	✓	✓	✓	✓
■ 6 Types 1/5	–	☐	☐	☐	☐	✓	✓	✓
☐ 7 Constants 0/1	–	–	–	–	–	–	–	✓
■ 8 Declarations and definitions 3/12	–	■	■	■	■	✓	✓	✓
■ 9 Initialization 2/3	–	☐	☐	☐	☐	✓	✓	✓
■ 10 Arithmetic type conversions 2/6	–	☐	☐	☐	☐	✓	✓	✓
■ 11 Pointer type conversions 4/5	–	■	■	■	■	✓	✓	✓
■ 12 Expressions 7/13	–	■	■	■	■	✓	✓	✓
■ 13 Control statement expressions 6/7	–	■	■	■	■	✓	✓	✓
■ 14 Control flow 4/10	–	■	■	■	■	✓	✓	✓

To make changes to the quality objectives settings, you must have a role of **Administrator**.

Previously, you could not see quality objective statistics for Bug Finder results. See Customize Software Quality Objectives.

Project Dashboard: Open results by clicking Dashboard charts

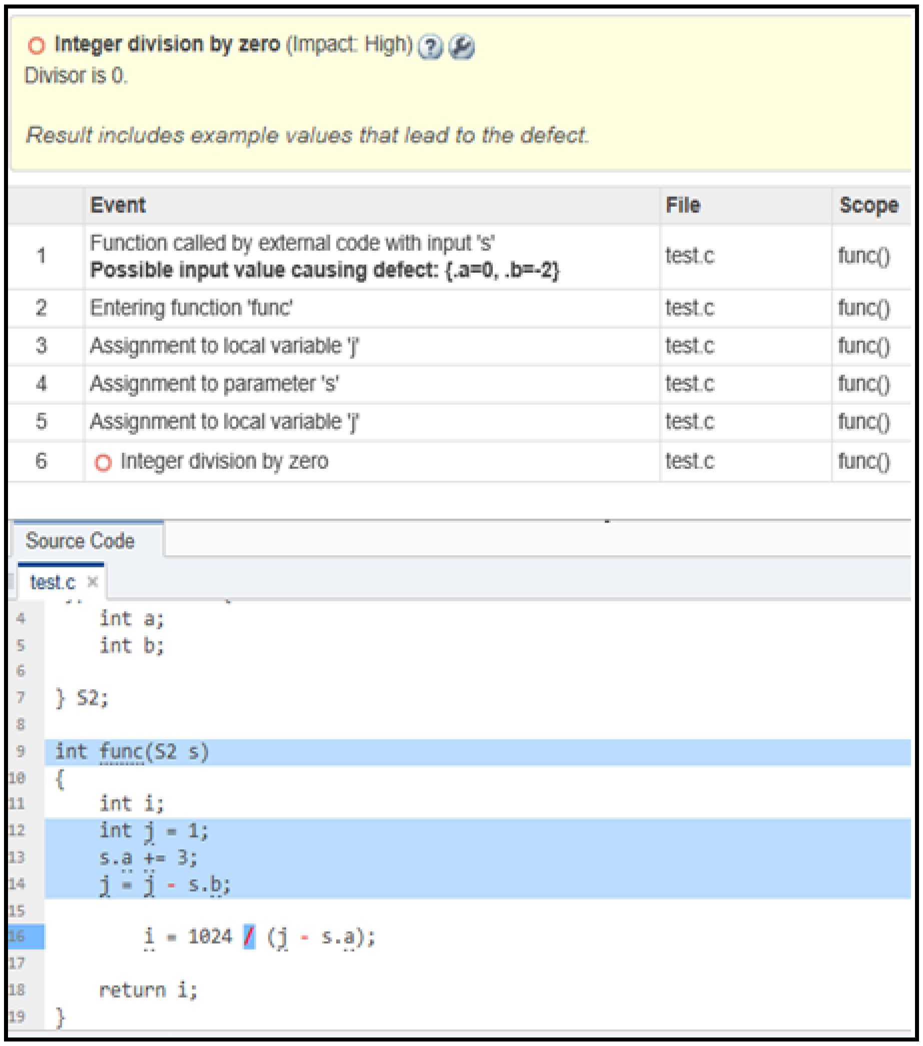
In R2020a, you can click a section of a pie chart or the legend of a pie chart to open the corresponding findings in the **Results List** and more easily narrow the scope of your review.



Family	ID	Type	Group
▼	517193...	Custom Rules	7 Functions
▼	517193...	Custom Rules	7 Functions
▽	517197...	MISRA C:2012	9 Initialization
▽	517194...	MISRA C:2012	14 Control statem
▼	517194...	Custom Rules	7 Functions
▼	517194...	Custom Rules	7 Functions
▼	517194...	Custom Rules	7 Functions
▼	517194...	Custom Rules	7 Functions
▼	517194...	Custom Rules	7 Functions

Extending Checkers: See example value for defect found with stricter analysis

Summary: In R2020a, if the analysis option **Run stricter checks considering all values of system inputs (-checks-using-system-input-values)** is enabled, for a subset of numerical and static memory defects, you can see an example of values that lead to the detected defect in the **Results Details**.



Integer division by zero (Impact: High)
Divisor is 0.

Result includes example values that lead to the defect.

	Event	File	Scope
1	Function called by external code with input 's' **Possible input value causing defect: {.a=0, .b=-2}**	test.c	func()
2	Entering function 'func'	test.c	func()
3	Assignment to local variable 'j'	test.c	func()
4	Assignment to parameter 's'	test.c	func()
5	Assignment to local variable 'j'	test.c	func()
6	Integer division by zero	test.c	func()

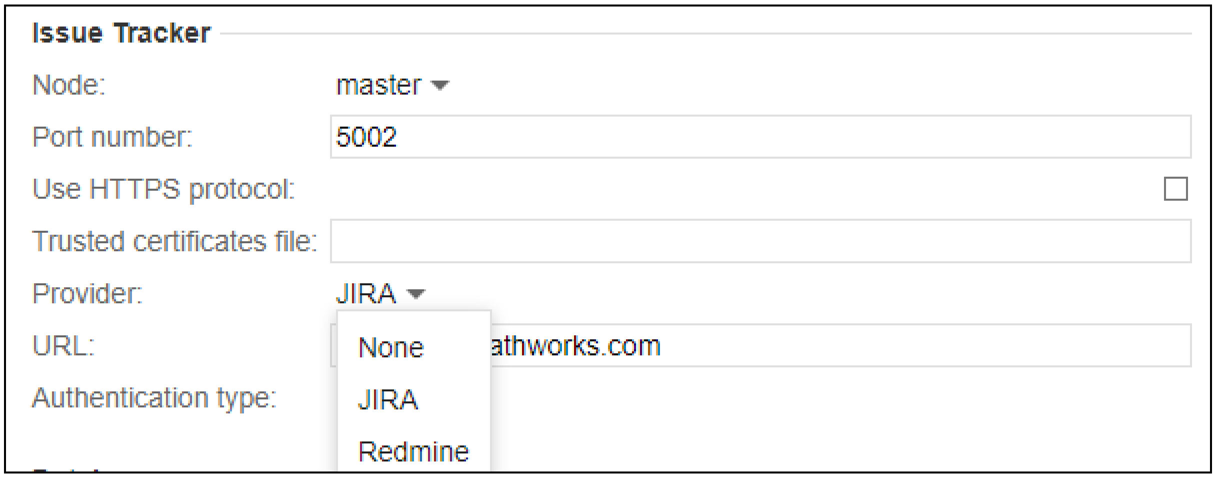
```
    int a;
    int b;

} S2;

int func(S2 s)
{
    int i;
    int j = 1;
    s.a += 3;
    j = j - s.b;

        i = 1024 / (j - s.a);

    return i;
}
```

You can use the example values to fix defects in your code that are due to specific system input values.

Installation

Installation and Configuration: New Issue Tracker service

In R2020a, use the new **Issue Tracker** service to configure Polyspace Access to integrate with the Jira software or Redmine bug tracking tools.



Issue Tracker

Node:	master
Port number:	5002
Use HTTPS protocol:	☐
Trusted certificates file:	
Provider:	JIRA
URL:	athworks.com
Authentication type:	

See Configure the **User Manager** and **Issue Tracker**.

Installation and Configuration: Change in default location of Polyspace Access data volume and working directories

In R2020a, the default location of the working directories of the Polyspace Access **Web Server** and **ETL** services and of the data volume is inside the folder where you unzipped the Polyspace Access ZIP file, under the `polyspace` folder.

Previously, the working directories of the **Web Server** and **ETL** were stored in the temporary files folder of your system (`/tmp` on Linux or `%TEMP%` on Windows). The data volume was stored under `/var/lib/docker/volumes` on Linux.

R2019b

Installation

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 Bug Finder™ Access™. Previously, you loaded all LDAP users listed under the **LDAP base** that you specified when you started Polyspace Bug Finder 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

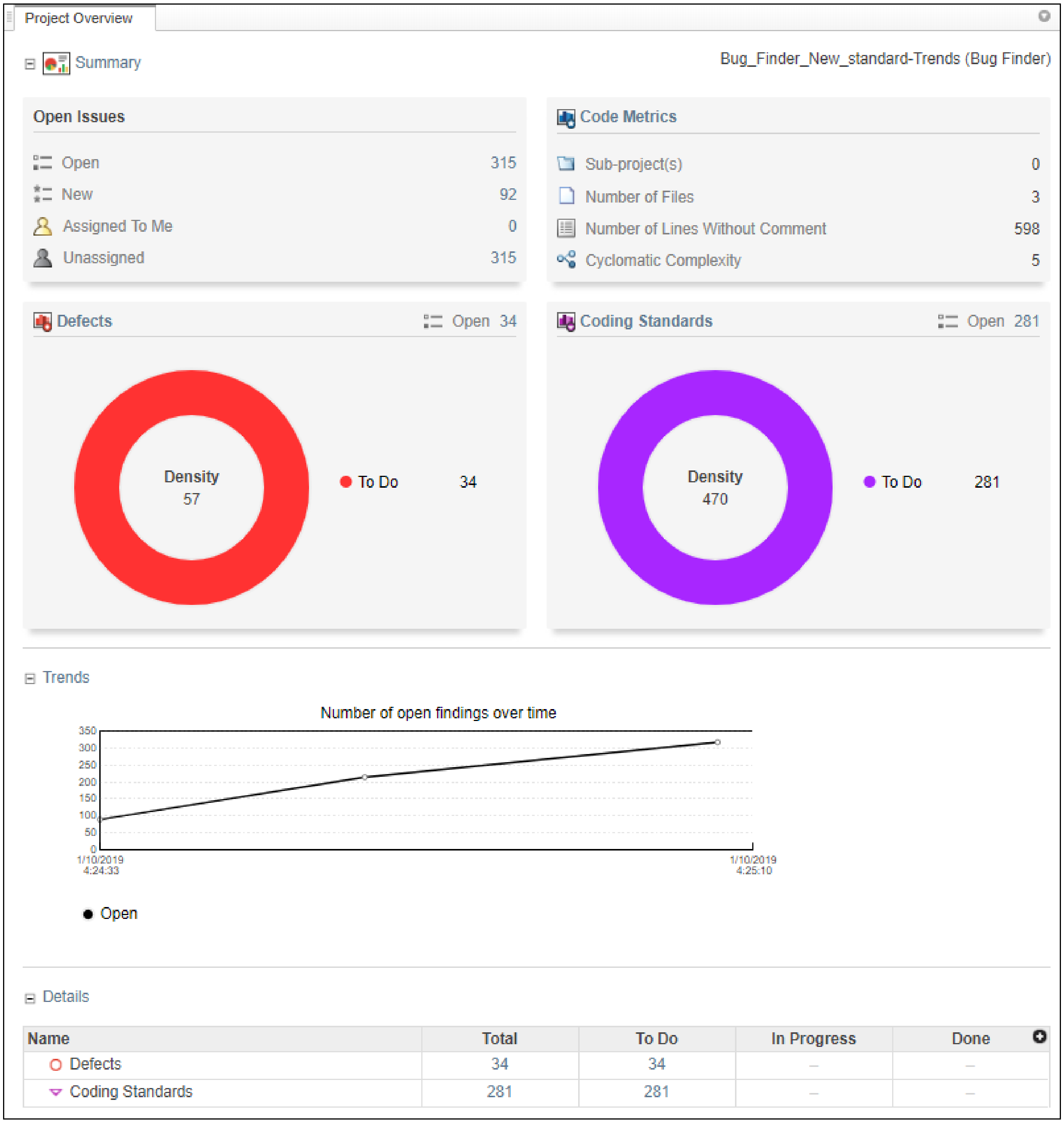
Version: 2.0

New Features

Dashboard and Review in Web Browser

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 Bug Finder Access **DASHBOARD**. When an analysis run is uploaded to the Polyspace Access database, the dashboard updates to give a snapshot of the findings, including a progress trend for number of findings compared to previous runs.



Project Overview

Summary

Bug_Finder_New_standard-Trends (Bug Finder)

Open Issues

Open	315
New	92
Assigned To Me	0
Unassigned	315

Code Metrics

Sub-project(s)	0
Number of Files	3
Number of Lines Without Comment	598
Cyclomatic Complexity	5

Defects Open 34

Density 57

● To Do 34

Coding Standards Open 281

Density 470

● To Do 281

Trends

Details

Name	Total	To Do	In Progress	Done
Defects	34	34	–	–
Coding Standards	281	281	–	–

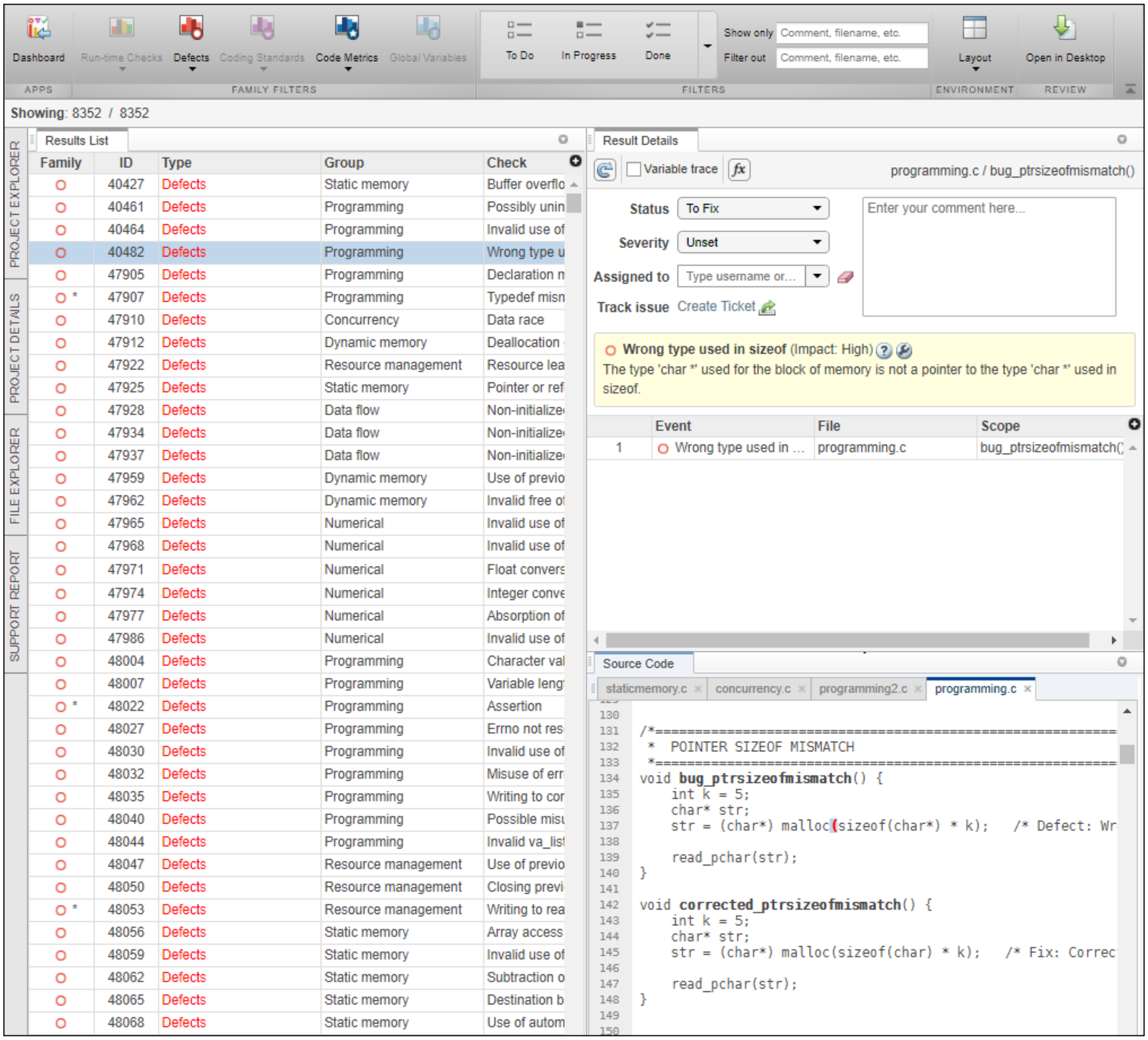
Additional Benefits:

- *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, move all the projects 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 Bug Finder Access license and the appropriate credentials can view the dashboard from their web browser.

Collaborative Review Support: Review Polyspace Bug Finder 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 Bug Finder Access **REVIEW** web interface. You do not need to install a Polyspace product on your machine to open and review analysis results.



Show only: Comment, filename, etc.
Filter out: Comment, filename, etc.

Showing: 8352 / 8352

Family	ID	Type	Group	Check
○	40427	Defects	Static memory	Buffer overflo
○	40461	Defects	Programming	Possibly unin
○	40464	Defects	Programming	Invalid use of
○	40482	Defects	Programming	Wrong type u
○	47905	Defects	Programming	Declaration n
○ *	47907	Defects	Programming	Typedef misn
○	47910	Defects	Concurrency	Data race
○	47912	Defects	Dynamic memory	Deallocation
○	47922	Defects	Resource management	Resource lea
○	47925	Defects	Static memory	Pointer or ref
○	47928	Defects	Data flow	Non-initialize
○	47934	Defects	Data flow	Non-initialize
○	47937	Defects	Data flow	Non-initialize
○	47959	Defects	Dynamic memory	Use of previo
○	47962	Defects	Dynamic memory	Invalid free of
○	47965	Defects	Numerical	Invalid use of
○	47968	Defects	Numerical	Invalid use of
○	47971	Defects	Numerical	Float convers
○	47974	Defects	Numerical	Integer conve
○	47977	Defects	Numerical	Absorption of
○	47986	Defects	Numerical	Invalid use of
○	48004	Defects	Programming	Character val
○	48007	Defects	Programming	Variable leng
○ *	48022	Defects	Programming	Assertion
○	48027	Defects	Programming	Errno not res
○	48030	Defects	Programming	Invalid use of
○	48032	Defects	Programming	Misuse of err
○	48035	Defects	Programming	Writing to cor
○	48040	Defects	Programming	Possible misu
○	48044	Defects	Programming	Invalid va_list
○	48047	Defects	Resource management	Use of previo
○	48050	Defects	Resource management	Closing previ
○ *	48053	Defects	Resource management	Writing to rea
○	48056	Defects	Static memory	Array access
○	48059	Defects	Static memory	Invalid use of
○	48062	Defects	Static memory	Subtraction o
○	48065	Defects	Static memory	Destination b
○	48068	Defects	Static memory	Use of autom

programming.c / bug_ptrsizeofmismatch()

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

Enter your comment here...

○ **Wrong type used in sizeof** (Impact: High)
The type 'char *' used for the block of memory is not a pointer to the type 'char *' used in sizeof.

	Event	File	Scope
1	○ Wrong type used in ...	programming.c	bug_ptrsizeofmismatch(

staticmemory.c × concurrency.c × programming2.c × programming.c ×

```
/*=============================================================
 *  POINTER SIZEOF MISMATCH
 *=============================================================
void bug_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char*) * k);   /* Defect: Wr

    read_pchar(str);
}

void corrected_ptrsizeofmismatch() {
    int k = 5;
    char* str;
    str = (char*) malloc(sizeof(char) * k);   /* Fix: Correc

    read_pchar(str);
}
```

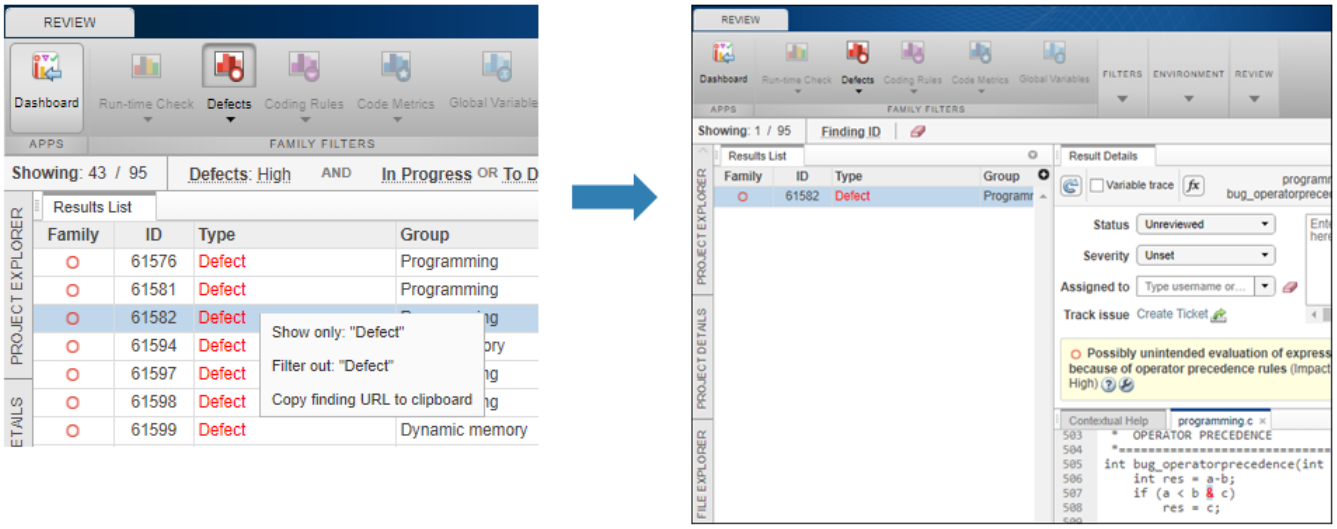
Additional Benefits:

- *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 Bug Finder Access license and the appropriate credentials can view the results from their web browser.

Collaborative Review Support: Share Polyspace Bug Finder results using web links

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



Showing: 43 / 95 Defects: High AND In Progress OR To D

Family	ID	Type	Group
○	61576	Defect	Programming
○	61581	Defect	Programming
○	61582	Defect	ng
○	61594	Defect	ory
○	61597	Defect	ng
○	61598	Defect	ng
○	61599	Defect	Dynamic memory

Showing: 1 / 95 Finding ID

Family	ID	Type	Group
○	61582	Defect	Programm

programm
bug_operatorprece

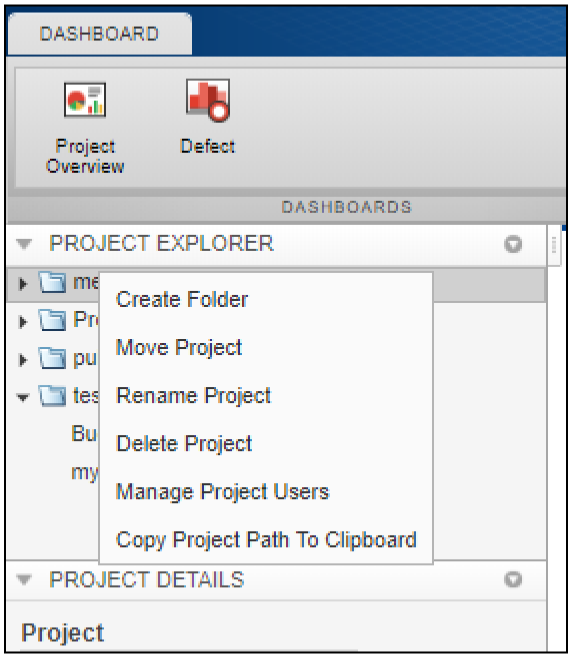
Status: Unreviewed
Severity: Unset
Assigned to: Type username or...
Track issue: Create Ticket

○ Possibly unintended evaluation of express because of operator precedence rules (Impact High)

```
 *   OPERATOR PRECEDENCE
 *=================================
int bug_operatorprecedence(int
    int res = a-b;
    if (a < b & c)
        res = c;
```

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

Summary:In R2019a, you can manage project users in Polyspace Bug Finder 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.

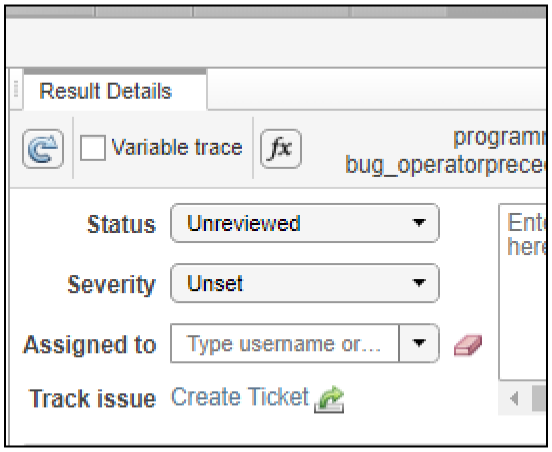


Additional Benefits:

- *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 Bug Finder results

Summary: In R2019a, Polyspace Bug Finder Access supports integration with the JIRA software. If you have an instance of the JIRA software, after you configure Polyspace Bug Finder Access, 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 Bug Finder Access web interface displays a link to the corresponding JIRA issue.