PolySpace® Release Notes

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "About Release Notes" on page 4.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version for C/C++ Products: V7.2 (R2010a)	Yes Details	Yes Summary	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	Printable Release Notes: PDF Current product documentation
Latest Version for Ada and Model Link Products: V5.5 (R2010a)	Yes Details	Yes Summary	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	Printable Release Notes: PDF Current product documentation

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V7.1 (R2009b) for C/C++ Products	Yes Details	Yes Summary	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	No
V5.4 (R2009b) for Ada and Model Link Products	Yes Details	Yes Summary	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	No
V7.0 (R2009a) for C/C++ Products	Yes Details	Yes Summary	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	No

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V5.3 (R2009a) for Ada and Model Link Products	Yes Details	No	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	No
V6.0 (R2008b) for C/C++ Products	Yes Details	No	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	No
V5.2 (R2008b) for Ada and Model Link Products	Yes Details	No	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports	No

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V5.1 (R2008a)	Yes Details	Yes Summary	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports	No
Previous Versions			Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports	No

About Release Notes

Use release notes when upgrading to a newer version to learn about new features and changes, and the potential impact on your existing files and practices. Release notes are also beneficial if you use or support multiple versions.

If you are not upgrading from the most recent previous version, review release notes for all interim versions, not just for the version you are installing. For example, when upgrading from V1.0 to V1.2, review the New Features and Changes, Version Compatibility Considerations, and Bug Reports for V1.1 and V1.2.

New Features and Changes

These include

- New functionality
- Changes to existing functionality
- Changes to system requirements (complete system requirements for the current version are at the MathWorks Web site)
- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, its description includes a **Compatibility Considerations** subsection that details the impact. For a list of all new features and changes that have reported compatibility impact, see the "Compatibility Summary for PolySpace Software" on page 83.

Compatibility issues that are reported after the product has been released are added to Bug Reports at the MathWorks Web site. Because bug fixes can sometimes result in incompatibilities, also review fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks Bug Reports is a user-searchable database of known problems, workarounds, and fixes. The MathWorks updates the Bug Reports database as new problems and resolutions become known, so check it as needed for the latest information.

Access Bug Reports at the MathWorks Web site using your MathWorks Account. If you are not logged in to your MathWorks Account when you link to Bug Reports, you are prompted to log in or create an account. You then can view bug fixes and known problems for R14SP2 and more recent releases.

Related Documentation at Web Site

Printable Release Notes (PDF). You can print release notes from the PDF version, located at the MathWorks Web site. The PDF version does not support links to other documents or to the Web site, such as to Bug Reports. Use the browser-based version of release notes for access to all information.

Product Documentation. At the MathWorks Web site, you can access complete product documentation for the current version and some previous versions, as noted in the summary table.

Version 7.2 (R2010a) PolySpace for C/C++ Products

This table summarizes what's new in V7.2 (R2010a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for C/C++ Product" on page 7
- "PolySpace® Server for C/C++ Product" on page 31

PolySpace Client for C/C++ Product

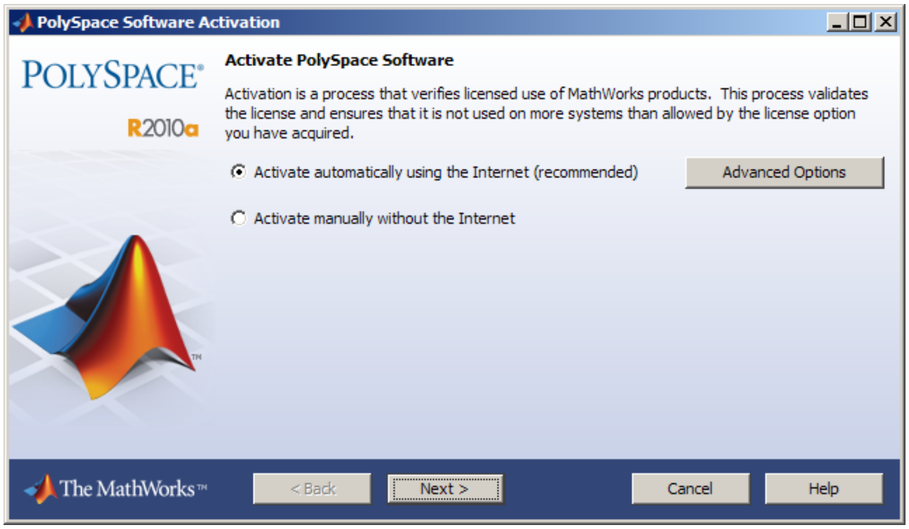
License Activation

PolySpace® products now support the MathWorks software activation mechanism.

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

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

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



Activate PolySpace Software

Activation is a process that verifies licensed use of MathWorks products. This process validates the license and ensures that it is not used on more systems than allowed by the license option you have acquired.

[✓ Activate automatically using the Internet (recommended)]

Activate manually without the Internet

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

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

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

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

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

MISRA C++ Checker

PolySpace software can now analyze your C++ code to check compliance with the MISRA® C++ coding standard.

The PolySpace MISRA C++ checker provides messages when MISRA C++ rules are not respected. Most messages are reported during the compile phase of a verification.

The MISRA C++ checker can check 167 of the 183 statically enforceable MISRA C++ coding rules.

Note The PolySpace MISRA C++ checker is based on MISRA C++:2008 – “Guidelines for the use of the C++ language in critical systems." For more information on these coding standards, see http://www.misra-cpp.com.

For more information, see “Checking Coding Rules”, in the *PolySpace Products for C++ User’s Guide*.

Source Code Comments

PolySpace software now allows you to place comments in your code that provide information about known coding rule violations and run-time errors. You can use these comments to

- Hide or highlight known coding rule violations.
- Highlight and categorize previously identified run-time errors.

This information can then make the review process quicker and easier by allowing you to focus on new coding rule violations and run-time errors. .

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

The coding rules log in the Launcher displays comments regarding coding rules. You can use these comments to filter out commented violations from the results, or simply to provide additional information on specific violations.

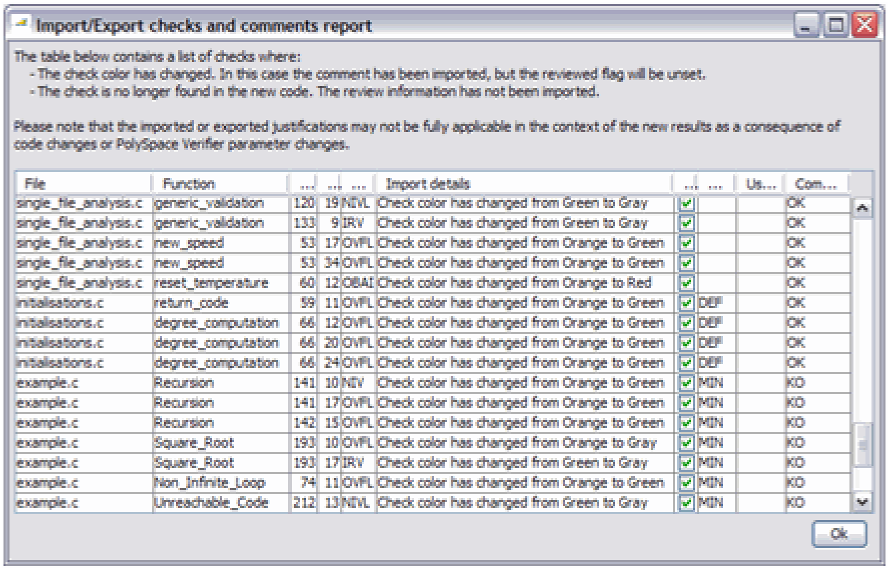
For more information, see “Highlighting Known Coding Rule Violations and Run-Time Errors” in the *PolySpace Products for C User’s Guide* or *PolySpace Products for C++ User’s Guide*.

Importing Review Comments

New Import/Export checks and comments report allows you to you to compare the source code and verification results from a previous verification to the current verification, and highlights differences in the results.

Importing review comments from a previous verification can be extremely useful, since it allows you to avoid reviewing checks twice, and to compare verification results over time. However, if your code has changed since the previous verification, or if you have upgraded to a new version of the software, the imported comments may not be applicable to your current results. For example, the color of a check may have changed, or the justification for an orange check may no longer be relevant to the current code.

Use the Import/Export checks and comments report to highlight these differences, and focus on unreviewed results.



Import/Export checks and comments report

The table below contains a list of checks where:
- The check color has changed. In this case the comment has been imported, but the reviewed flag will be unset.
- The check is no longer found in the new code. The review information has not been imported.

Please note that the imported or exported justifications may not be fully applicable in the context of the new results as a consequence of code changes or PolySpace Verifier parameter changes.

File	Function	...	...	...	Import details	...	...	Us...	Com...
single_file_analysis.c	generic_validation	120	19	NIVL	Check color has changed from Green to Gray	✓			OK
single_file_analysis.c	generic_validation	133	9	IRV	Check color has changed from Green to Gray	✓			OK
single_file_analysis.c	new_speed	53	17	OVFL	Check color has changed from Orange to Green	✓			OK
single_file_analysis.c	new_speed	53	34	OVFL	Check color has changed from Orange to Green	✓			OK
single_file_analysis.c	reset_temperature	60	12	OBAI	Check color has changed from Orange to Red	✓			OK
initialisations.c	return_code	59	11	OVFL	Check color has changed from Orange to Green	✓	DEF		OK
initialisations.c	degree_computation	66	12	OVFL	Check color has changed from Orange to Green	✓	DEF		OK
initialisations.c	degree_computation	66	20	OVFL	Check color has changed from Orange to Green	✓	DEF		OK
initialisations.c	degree_computation	66	24	OVFL	Check color has changed from Orange to Green	✓	DEF		OK
example.c	Recursion	141	10	NIV	Check color has changed from Orange to Green	✓	MIN		KO
example.c	Recursion	141	17	OVFL	Check color has changed from Orange to Green	✓	MIN		KO
example.c	Recursion	142	15	OVFL	Check color has changed from Orange to Green	✓	MIN		KO
example.c	Square_Root	193	10	OVFL	Check color has changed from Orange to Gray	✓	MIN		KO
example.c	Square_Root	193	17	IRV	Check color has changed from Green to Gray	✓	MIN		KO
example.c	Non_Infinite_Loop	74	11	OVFL	Check color has changed from Orange to Green	✓	MIN		KO
example.c	Unreachable_Code	212	13	NIVL	Check color has changed from Green to Gray	✓	MIN		KO

For more information, see "Importing and Exporting Review Comments" in the *PolySpace Products for C User's Guide*.

Compatibility Considerations. In previous releases, when you specified the option `-keep-all-files`, it was possible to add comments to the results for a specific verification level (for example, `pass2`) , and then import them into another set of results (for example `pass4`) in the same results folder.

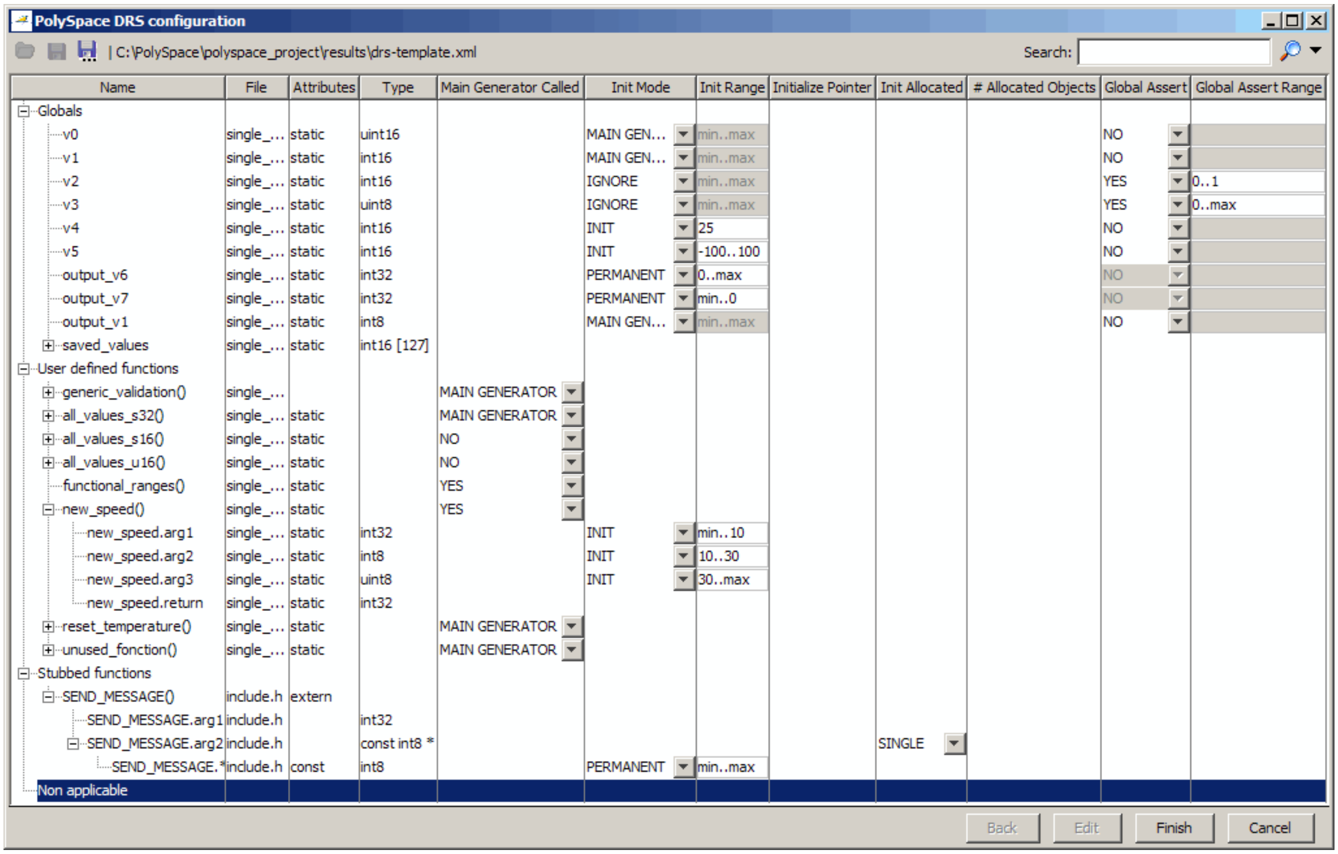
This is no longer possible in R2010a.

Data Range Specifications (DRS) Enhancements

Enhanced Data Range Specifications, including new format and workflow.

The PolySpace Data Range Specifications (DRS) feature now allows you to set constraints on data ranges using a new graphical user interface. When you enable the DRS feature, PolySpace software analyzes the files in your project, and generate a DRS template containing all the global variables, user defined functions, and stub functions for which you can specify data ranges.

To specify data ranges, you then edit this template using the PolySpace DRS configuration interface.



PolySpace DRS configuration

| C:\PolySpace\polyspace_project\results\drs-template.xml   Search:

Name	File	Attributes	Type	Main Generator Called	Init Mode	Init Range	Initialize Pointer	Init Allocated	# Allocated Objects	Global Assert	Global Assert Range
Globals											
v0	single_…	static	uint16		MAIN GEN…	min..max				NO	
v1	single_…	static	int16		MAIN GEN…	min..max				NO	
v2	single_…	static	int16		IGNORE	min..max				YES	0..1
v3	single_…	static	uint8		IGNORE	min..max				YES	0..max
v4	single_…	static	int16		INIT	25				NO	
v5	single_…	static	int16		INIT	-100..100				NO	
output_v6	single_…	static	int32		PERMANENT	0..max				NO	
output_v7	single_…	static	int32		PERMANENT	min..0				NO	
output_v1	single_…	static	int8		MAIN GEN…	min..max				NO	
saved_values	single_…	static	int16 [127]								
User defined functions											
generic_validation()	single_…			MAIN GENERATOR							
all_values_s32()	single_…	static		MAIN GENERATOR							
all_values_s16()	single_…	static		NO							
all_values_u16()	single_…	static		NO							
functional_ranges()	single_…	static		YES							
new_speed()	single_…	static		YES							
new_speed.arg1	single_…	static	int32		INIT	min..10					
new_speed.arg2	single_…	static	int8		INIT	10..30					
new_speed.arg3	single_…	static	uint8		INIT	30..max					
new_speed.return	single_…	static	int32								
reset_temperature()	single_…	static		MAIN GENERATOR							
unused_fonction()	single_…	static		MAIN GENERATOR							
Stubbed functions											
SEND_MESSAGE()	include.h	extern									
SEND_MESSAGE.arg1	include.h		int32								
SEND_MESSAGE.arg2	include.h		const int8 *					SINGLE			
SEND_MESSAGE.*	include.h	const	int8		PERMANENT	min..max					
Non applicable											

In addition, the DRS feature now allows you to specify constraints for additional types of data, including:

- Input parameters for user-defined functions called by the main generator
- Static variables
- Pointers (C only)

For more information, see “Specifying Data Ranges for Variables and Functions (Contextual Verification)” in the *PolySpace Products for C User’s Guide* or *PolySpace Products for C++ User’s Guide*.

Compatibility Considerations. Symbols ranged by DRS (`init`, `permanent` or `globalassert` mode) are no longer ignored by the main-generator. This can lead to differences in values and colors, for example full range instead of 0, or orange instead of green.

Pointer Information in the Viewer

Enhanced ToolTips in the Viewer now display pointer information, in addition to data ranges.

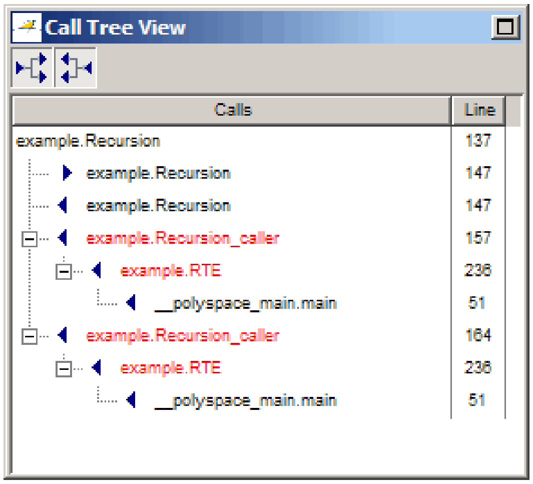
The software now provides, through tooltip messages, useful information about pointers to variables or functions. You see this information in the source code view when you place your cursor over a pointer, dereference character, function call, or function declaration. In addition, if you click a pointer check, dereference character, function call, or function declaration, the software displays pointer information in the selected check view.

For more information, see “Using Pointer Information in the Viewer” in the *PolySpace Products for C User’s Guide* or *PolySpace Products for C++ User’s Guide*.

Enhanced Call Tree View and Variables View (Data Dictionary)

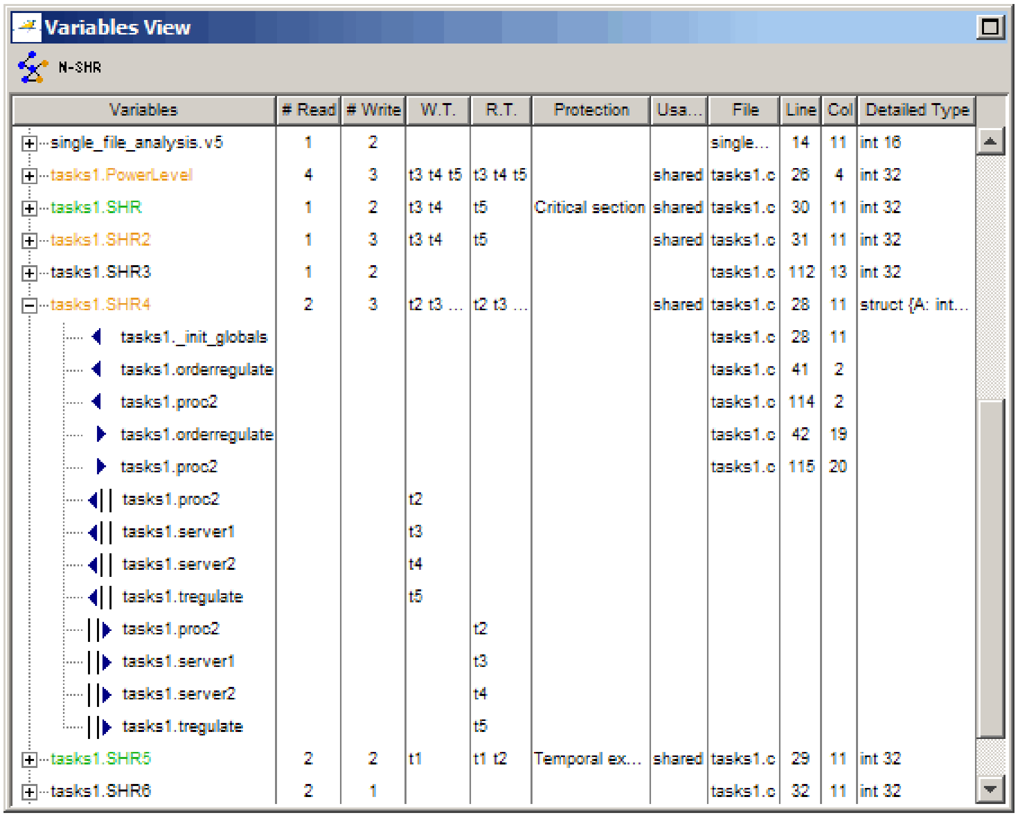
Enhanced user interface of the Call Tree View and Variables View improves navigation and usability.

In the Call Tree View, you can now double click any function call to go directly to the function definition.



Calls	Line
example.Recursion	137
example.Recursion	147
example.Recursion	147
example.Recursion_caller	157
example.RTE	236
__polyspace_main.main	51
example.Recursion_caller	164
example.RTE	236
__polyspace_main.main	51

In the Variables View, you can now right-click a variable to show legend information, and can open the concurrent access graph for a variable directly from the Variables View.



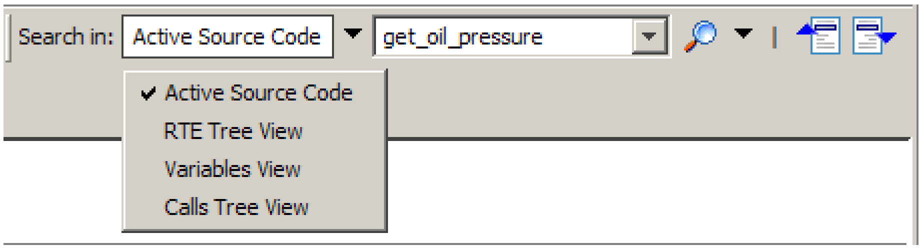
Variables	# Read	# Write	W.T.	R.T.	Protection	Usa...	File	Line	Col	Detailed Type
single_file_analysis.v5	1	2					single...	14	11	int 16
tasks1.PowerLevel	4	3	t3 t4 t5	t3 t4 t5		shared	tasks1.c	26	4	int 32
tasks1.SHR	1	2	t3 t4	t5	Critical section	shared	tasks1.c	30	11	int 32
tasks1.SHR2	1	3	t3 t4	t5		shared	tasks1.c	31	11	int 32
tasks1.SHR3	1	2					tasks1.c	112	13	int 32
tasks1.SHR4	2	3	t2 t3 ...	t2 t3 ...		shared	tasks1.c	28	11	struct {A: int...
◀ tasks1._init_globals							tasks1.c	28	11	
◀ tasks1.orderregulate							tasks1.c	41	2	
◀ tasks1.proc2							tasks1.c	114	2	
▶ tasks1.orderregulate							tasks1.c	42	19	
▶ tasks1.proc2							tasks1.c	115	20	
◀|| tasks1.proc2			t2							
◀|| tasks1.server1			t3							
◀|| tasks1.server2			t4							
◀|| tasks1.tregulate			t5							
||▶ tasks1.proc2				t2						
||▶ tasks1.server1				t3						
||▶ tasks1.server2				t4						
||▶ tasks1.tregulate				t5						
tasks1.SHR5	2	2	t1	t1 t2	Temporal ex...	shared	tasks1.c	29	11	int 32
tasks1.SHR6	2	1					tasks1.c	32	11	int 32

For more information, see "Exploring the Viewer Window" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Enhanced Search Function in Viewer

Enhanced Search feature in the Viewer improves navigation in your results.

The Viewer toolbar now contains a Search interface. This allows you to quickly enter search terms, specify search options, and set the scope for your search.



For more information, see "Exploring the Viewer Window" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Filtering Orange Checks in Viewer (C only)

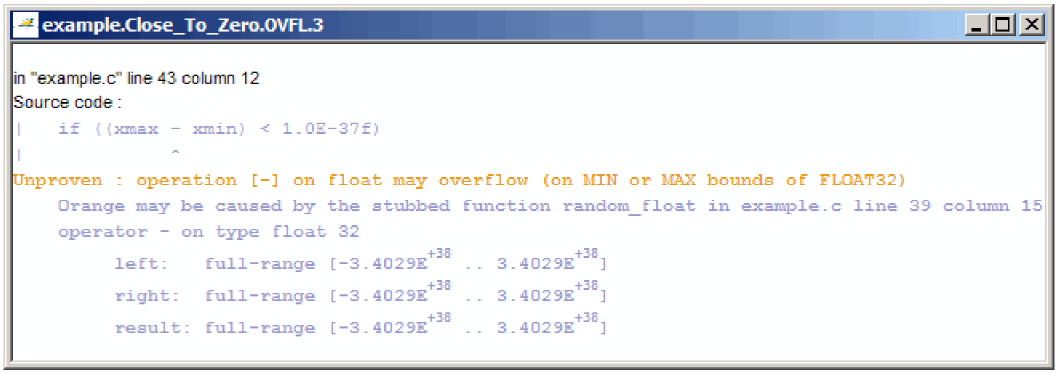
PolySpace verification now identifies orange checks caused by input data. The software provides additional information on these orange checks, and allows you to hide them in the Viewer.

Note Although this type of orange check could reveal a bug, they usually do not.

Verification can identify orange checks caused by:

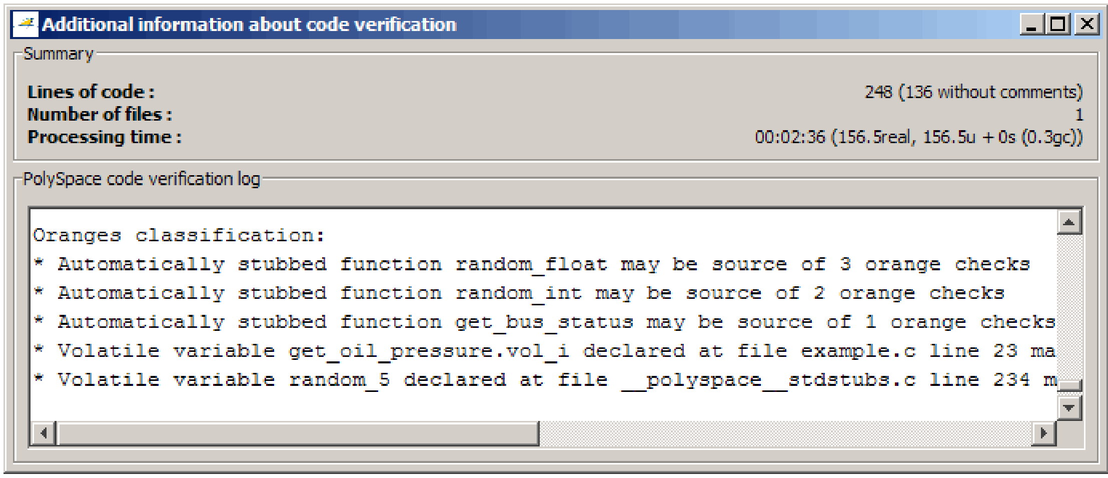
- Stubs
- Main-generator calls
- Volatile variables
- Extern variables
- Absolute address

When the software identifies this type of orange check, the Viewer provides information on its cause.



```
in "example.c" line 43 column 12
Source code :
|    if ((xmax - xmin) < 1.0E-37f)
|                ^
Unproven : operation [-] on float may overflow (on MIN or MAX bounds of FLOAT32)
    Orange may be caused by the stubbed function random_float in example.c line 39 column 15
    operator - on type float 32
         left:   full-range [-3.4029E+38 .. 3.4029E+38]
         right:  full-range [-3.4029E+38 .. 3.4029E+38]
         result: full-range [-3.4029E+38 .. 3.4029E+38]
```

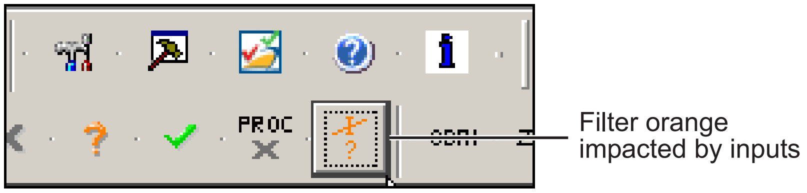
The PolySpace code verification log file also lists possible sources of imprecision for orange checks.


Additional information about code verification
Summary
PolySpace code verification log

Lines of code :	248 (136 without comments)
Number of files :	1
Processing time :	00:02:36 (156.5real, 156.5u + 0s (0.3gc))

```
Oranges classification:
* Automatically stubbed function random_float may be source of 3 orange checks
* Automatically stubbed function random_int may be source of 2 orange checks
* Automatically stubbed function get_bus_status may be source of 1 orange checks
* Volatile variable get_oil_pressure.vol_i declared at file example.c line 23 ma
* Volatile variable random_5 declared at file __polyspace__stdstubs.c line 234 m
```

In addition, you can now hide these types of orange checks in the Viewer. When using Expert mode, click the filter button to hide oranges impacted by input data.



For more information, see “Filtering Checks” in the *PolySpace Products for C User’s Guide* or *PolySpace Products for C++ User’s Guide*.

Methodological Assistant Enhancements

Enhanced Methodological Assistant in the Viewer.

The Methodological Assistant now allows you to define either a minimum percentage of orange checks to review, or a specific number of orange checks to review. This makes it easier to set specific quality criteria for your code at each level of review.

In addition, the Methodological Assistant now presents checks in a more logical order. Checks that are most likely to reveal bugs appear first, while non-useful checks no longer appear.

The new order of checks is:

1. All red checks (an error always occurs)
2. Orange checks known to produce errors in some situations (dark orange). For example, red for one call to a procedure and green for another.
3. Some gray checks (UNR checks)
4. Other orange checks (depending on the methodology and criterion level)

Most gray checks no longer appear in the Methodological Assistant, since reviewing many gray checks that occur after a red check is not useful. Only UNR checks that are not nested within dead code blocks appear in assistant mode.

Compatibility Considerations. The number of checks presented for review in Assistant mode is different than in previous releases, since most gray checks no longer appear. In addition, the order in which you review checks is different.

Class Analyzer Enhancements for C++

Enhanced class analyzer can analyze a file with more than one class.

Unit-by-unit verifications can now verify files containing more than one class. Every class and function out of class contained in such files is now verified.

For more information, see "PolySpace Class Analyzer" in the *PolySpace Products for C++ User's Guide*.

Compatibility Considerations. In `-unit-by-unit` mode, files that previously were not verified because they contained more than one class are now verified.

Change to Time Format in Log File

The time format reported in the log file has been updated to provide more information.

Example of new line (R2010a and later):

```
User time for polyspace-c:  00:02:24 (144.6real, 144.6u + 0s
(0.3gc))
```

Example of old line (R2009b and earlier):

```
User time for rte-kernel:  4684.4real, 4319.2u + 324.6s (0.3gc)
```

Compatibility Considerations. The new time format can impact some scripts that summarize information from the log file.

Merging of `OVFL` and `UNFL` Checks

Overflow (`OVFL`) and underflow (`UNFL`) checks have been merged into a single `OVFL` check. This reduces the number of orange checks you need to review, while continuing to provide the same information.

For red and orange checks, the check message provides the bounds that cause the overflow.

Compatibility Considerations. The Selectivity rate of your results may change when compared to previous versions of the software. Underflows and overflows are now identified as a single check, so the Selectivity will decrease if the checks were green (2 green checks become 1 green), but will increase if the checks were both orange (2 orange checks become 1 orange).

Improved `UNR` Checks

Enhanced unreachable code (UNR) checks now provide additional information to help you understand the results. UNR checks now include information on:

- Localization of condition
- Type of condition
- End of block localization

For example:

```
// UNR (unreachable code) => UNR (unreachable code)   \
(end of block at line YYY)

// UNR (unreachable code) => UNR (unreachable code)   \
(condition at line XXX, column AAA) ?
```

In addition, verification now reports new `UNR` checks on:

- unreachable statements after `return`, `break`, `goto`, and `continue` statements.
- `if` statements when the `if` condition is always true and if there is no `else` statement.

For more information on these new checks, see "Changes to Verification Results" on page 21.

Compatibility Considerations. The number of checks in your verification results may change due to the new `UNR` checks.

Changes to Verification Results

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

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

Merging of OVFL and UNFL Checks. Overflow (OVFL) and underflow (UNFL) checks have been merged into a single OVFL check. This reduces the number of orange checks you need to review, while continuing to provide the same information.

For red and orange checks, the check message provides the bounds that cause the overflow.

The Selectivity rate of your results may change when compared to previous versions of the software.

New Gray (`UNR`) Checks on `return`, `break`, `goto`, and `continue` Statements. Verification now reports gray `UNR` checks on unreachable statements after `return`, `break`, `goto`, and `continue` statements.

For example:

```
switch (counter) {
 case 0:
  counter = 0;
  break;
case 1:
 counter = 2;
 break;
  counter = 2; /* unreachable code ! */
  break;
```

The number of checks in your verification results may increase due to these new UNR checks.

New Gray (`UNR`) Check on If Statement Without Else. Verification now reports a gray `UNR` check on an `if` statement when the `if` condition is always true and if there is no `else` statement.

This allows you to find `if` branches that are always reachable, even when there is no `else`.

For example:

```
if (true())  // UNR if-condition always evaluates to true
```

```
   {
    // ...
   }
```

The number of checks in your verification results may increase due to new UNR checks.

Nested Gray (UNR) Checks No Longer Appear in Reports. Nested UNR checks in unreachable blocks no longer appear in the Methodological Assistant, or in generated reports.

The number of checks in generated reports may decrease due to elimination of these checks..

Dead Code on Else Branch. Verification now reports gray `UNR` checks on empty branches.

For example:

```
void fct (void)
{
 int a = 1;
 if (a){
  a++;
 }
 else // ==> Now gray UNR
 {
 // dummy
 }
}
```

The number of checks in your verification results may increase due to the new UNR check on empty branches.

Data Ranges for Fields of Structures (C). Symbols ranged by DRS (`init`, `permanent` or `globalassert` mode) are now considered by the main-generator.

In previous releases, if DRS provided ranges for some fields of a structure, the other fields (not ranged by DRS) were not initialized by the main-generator, and therefore had an initial value of 0.

For example:

```
// DRS: s.x 0 10 init

struct { int x; int y; } s;
int foo(void)
{
  return s.y; // y value: 0, full-range expected
}
```

Symbols ranged by DRS are no longer ignored by the main-generator. This can lead to differences in values and colors, for example full range instead of 0, or orange instead of green.

Functions Called Before Main in Unit-by-unit Verification (C++). The behavior of the option `-function-called-before-main` has changed for unit-by-unit verifications of C++ code.

When you set the option `-function-called-before-main` in unit-by-unit mode:

- If the `init` function is an out of class function, it is called at the beginning of the generated main (before calls to constructors).
- If the `init` function is a method, it is called after all constructor calls of the corresponding class.

In previous releases, the `init` function was always called after constructor calls for each class.

Verification results may change when compared to previous versions of the software, due to changes in the call sequence.

Main Generator Initialization of Function Pointers. The main-generator now initializes function pointers with default-mode stubs instead of pure stubs.

In previous releases, the main-generator initialized function pointers with pointers to pure functions.

This change may lead to differences in the color of checks in your results. For example:

```
int x;
  s->fptr(&x);
  read(x); // LNIV red with 9b -> orange with 10a
```

OVFL Check on Array Index Removed. In previous releases, verification reported an overflow (OVFL) check on pointer/array dereference. However, this overflow never occurred if there was an OBAI problem first. Therefore, the check was not useful.

In R2010a, the OVFL check no longer appears on array index, the check has been merged into the OBAI check.

For example, in the following code there is no OVFL check on the array index.

```
int main(void)
{
   volatile int i,x;
   int tab[10];

   x = tab[i];

}
```

The Selectivity of your results may change when compared to previous versions of the software. The OVFL check on array access has been merged into the OBAI check, so there are fewer checks reported. Selectivity will increase if the overflow check was orange, but will decrease if the OVFL check was green.

IDP Check on Local Member Access Removed (C++). Verification no longer reports an IDP check on local member access.

In previous releases, verification reported an IDP check. This IDP appeared on the "`.`" when accessing the field of an object returned by copy construction.

For example:

```
struct C {
  C(const C&c1) { k =c1.k; }
  C() { k = 0 ;}
  int k ;
} ;

C g() {
  C ret ;
  ret.k = 2 ; // IDP on "." here
  return ret;
}

int main() {
  C c = g() ;
}
```

However, this check was caused by an internal pointer and was not useful.

The Selectivity of your results may change when compared to previous versions of the software.

OBAI Check on Dynamic Initialization of Array Removed (C++).

Verification no longer reports an OBAI check on dynamic initialization of array.

In previous releases, verification reported an OBAI check. The OBAI check appeared on dynamic initialization of array with an aggregate.

For example:

```
nt main(void)
{
  float tab[] = // extra green obai check
    {
      4.3,
      0.0F
    };

  return 0;
}
```

However, this check was caused by an internal translation and was not useful.

The Selectivity of your results may change when compared to previous versions of the software.

Duplicate Checks in `For/While` Loops Removed. Verification no longer reports duplicate checks in condition expression of `for` and `while` loops.

Any duplicate checks on a loop condition are now merged in a single check, except when condition expression is complex.

Due to the reduction in the number of checks, the selectivity of your results may change when compared to previous versions of the software.

`malloc(0)` Limitation Removed. Verification no longer has a limitation when `malloc(0)` returns a null pointer.

In previous releases, verification reported a green check on the following code:

```
assert(malloc(0) == NULL) ;
```

However, this construction could fail. The software now correctly verifies this construction.

Verification results may change when compared to previous versions of the software.

Change in OOP on Deletion of Null Pointer (C++). Verification no longer reports a red OOP check when deleting a null pointer.

In previous releases, verification reported a red OOP check on the following code:

```
struct A {
  virtual void f() { }
  ~A() { }
} ;

int main() {
  A* pa ;
  if (0) pa = (A*) 0Xfff;
```

```
    pa = 0;
    delete pa ; // red OOP
  }
```

However, calling "delete" on a null pointer is allowed. The red OOP when deleting a null pointer is now gray.

Verification results may change when compared to previous versions of the software.

Change to IDP Check When Accessing a Field of an Inherited Class (C). Verification no longer reports a red IDP check when accessing a field of an inherited class with an mcpu target.

In previous releases, verification reported a red IDP check.

For example:

```
struct Val {
 int val;
};

struct Left : virtual Val {
 int left;
 virtual int get_left() { return left; } // polymorphic:yes
};

struct Right : virtual Val {
 int right;
 virtual int get_right() { return right; }
};

struct S : Left, Right {      // multiple:yes
};

S s = S();
Left& le = s;         // intermediate:global, reference:yes
Right& re = s;         // intermediate:global, reference:yes

int main(void){
    assert(re.val == 0);    // Unexpected red IDP
  }
```

However, this was not actually an error. The check is no longer red.

The color of the IDP check has changed when compared to previous versions of the software.

Changes to Coding Rules Checker Results

- “Compatibility Considerations” on page 29
- “MISRA-C Rule 10.1 Violations on Constant Operands” on page 29
- “MISRA-C Rule 12.5 Violation Report Improved” on page 30
- “MISRA-C Rule 7.1 Violations on File Names of Preprocessed Files” on page 30
- “MISRA-C Rule 5.4 Violations on Anonymous Structures and Unions” on page 30
- “JSF Rule AV-151 Violations on Evaluation of Constant” on page 30

Compatibility Considerations. Due to changes in the coding rules checker, the number of coding rule violations may change when compared to previous versions of the software.

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

MISRA-C Rule 10.1 Violations on Constant Operands. The MISRA-C checker no longer reports errors for rule 10.1, “The value of an expression of integer type shall not be implicitly converted to a different underlying type,” for certain constructions. For example:

```
int i;
for (i = 0; i < 12; i++)
```

An integer constant that fits into the size of a `char` is now seen as a `signed char` whatever the sign of `char` (this depends on the selected target or is set by option).

If you use the options `-target powerpc` or `-default-sign-of-char unsigned`, the coding rules checker will report fewer violations of MISRA-C rule 10.1 on constant operands.

MISRA-C Rule 12.5 Violation Report Improved. The coding rules checker now reports a column number for violations of MISRA-C rule 12.5.

You may see more violations of rule 12.5, since two violations that occur on same line but in different columns are now identified separately.

MISRA-C Rule 7.1 Violations on File Names of Preprocessed Files. The coding rules checker no longer reports violations of MISRA-C rule 7.1 on the names of internal preprocessing files. These violations occurred in projects containing Japanese characters.

You may see fewer violations of rule 7.1 in MISRA reports.

MISRA-C Rule 5.4 Violations on Anonymous Structures and Unions. The coding rules checker no longer reports violations of MISRA-C rule 5.4 on anonymous struct/union fields.

You may see fewer violations of rule 5.4 in MISRA reports.

JSF Rule AV-151 Violations on Evaluation of Constant. The coding rules checker no longer reports violations of JSF rule AV-151 on internal evaluation of a constant value, for example when there is an expression in an enum list.

You may see fewer violations of rule AV-151 in JSF reports.

Enumerated Types Support

The option `-enum-type-definition` allows verification to use different base types to represent an enumerated type, depending on the enumerator values and the selected definition.

When using this option, each enum type is represented by the smallest integral type that can hold all its enumeration values.

Possible values are:

- `defined-by-standard`.
- `auto-signed-first`.
- `auto-unsigned-first`

For more information, see "Assumptions" in the *PolySpace Products for C Reference* or *PolySpace Products for C++ Reference*.

New Target Processor Support

Added support for the `c18` 24-bit target processor (C only).

For more information, see "Predefined Target Processor Specifications" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Operating System Support

Added support for the following Linux® distributions:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for C/C++ Product

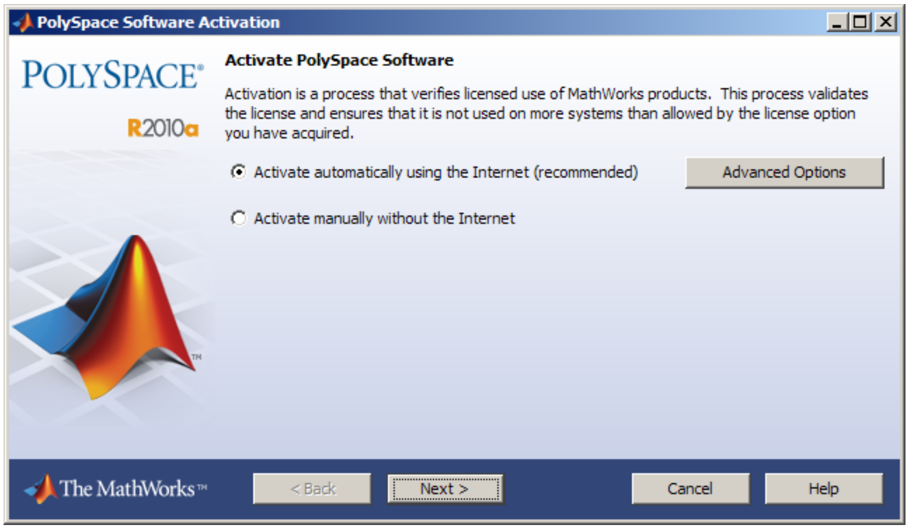
License Activation

PolySpace products now support the MathWorks software activation mechanism.

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

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

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



Activate PolySpace Software

Activation is a process that verifies licensed use of MathWorks products. This process validates the license and ensures that it is not used on more systems than allowed by the license option you have acquired.

[✓ Activate automatically using the Internet (recommended)]

Activate manually without the Internet

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

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

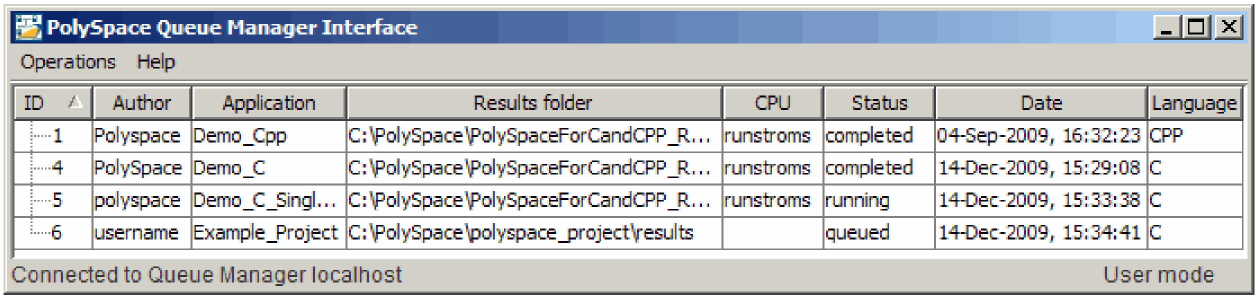
If your PolySpace system is not connected to the internet, you can access the MathWorks License Center on a computer with internet access, activate your license, and download a license file for transfer to your PolySpace system. If you do not have access to a computer with an Internet connection, contact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

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

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

Queue Manager Interface

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



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

Connected to Queue Manager localhost   User mode

For more information, see "Managing Verification Jobs Using the PolySpace Queue Manager"in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Operating System Support

Added support for the following Linux distributions:

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

For more information, see the *PolySpace Installation Guide*.

Version 5.5 (R2010a) PolySpace for Ada and Model Link Products

This table summarizes what's new in V5.5 (R2010a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for Ada Product" on page 34
- "PolySpace® Server for Ada Product" on page 37
- "PolySpace Model Link SL Product" on page 39

PolySpace Client for Ada Product

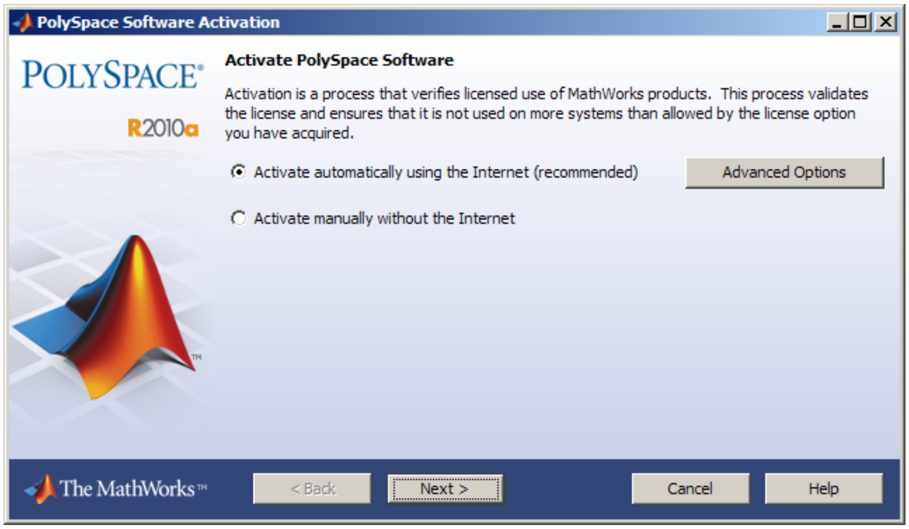
License Activation

PolySpace products now support the MathWorks software activation mechanism.

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

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

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



Activate PolySpace Software

Activation is a process that verifies licensed use of MathWorks products. This process validates the license and ensures that it is not used on more systems than allowed by the license option you have acquired.

[✓ Activate automatically using the Internet (recommended)]

Activate manually without the Internet

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

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

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

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

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

Source Code Comment Support

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

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

For more information, see “Highlighting Known Run-Time Errors” in the *PolySpace Products for Ada User’s Guide*.

Eclipse Integration

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

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

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

- A contextual menu that allows you to launch a verification of one or more files.

- Views in the Eclipse editor that allow you to set verification parameters and monitor verification progress.

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

Operating System Support

Added support for the following Linux distributions:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for Ada Product

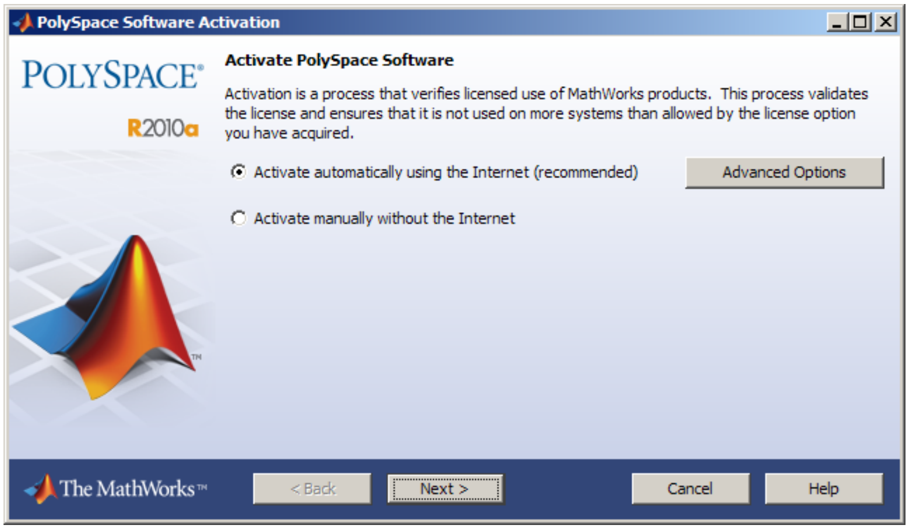
License Activation

PolySpace products now support the MathWorks software activation mechanism.

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

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

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


PolySpace Software Activation
POLYSPACE®
R2010a
Activate PolySpace Software
Activation is a process that verifies licensed use of MathWorks products. This process validates the license and ensures that it is not used on more systems than allowed by the license option you have acquired.
[✓ Activate automatically using the Internet (recommended)]
Advanced Options
Activate manually without the Internet
The MathWorks™
< Back
Next >
Cancel
Help

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

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

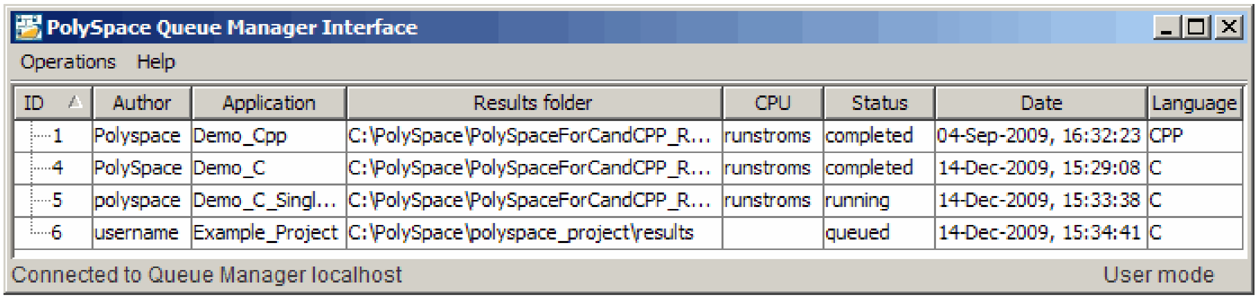
If your PolySpace system is not connected to the internet, you can access the MathWorks License Center on a computer with internet access, activate your license, and download a license file for transfer to your PolySpace system. If you do not have access to a computer with an Internet connection, contact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

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

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

Queue Manager Interface

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



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

Connected to Queue Manager localhost   User mode

For more information, see “Managing Verification Jobs Using PolySpace Queue Manager”in the *PolySpace Products for Ada User’s Guide.*

Operating System Support

Added support for the following Linux distributions:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link SL Product

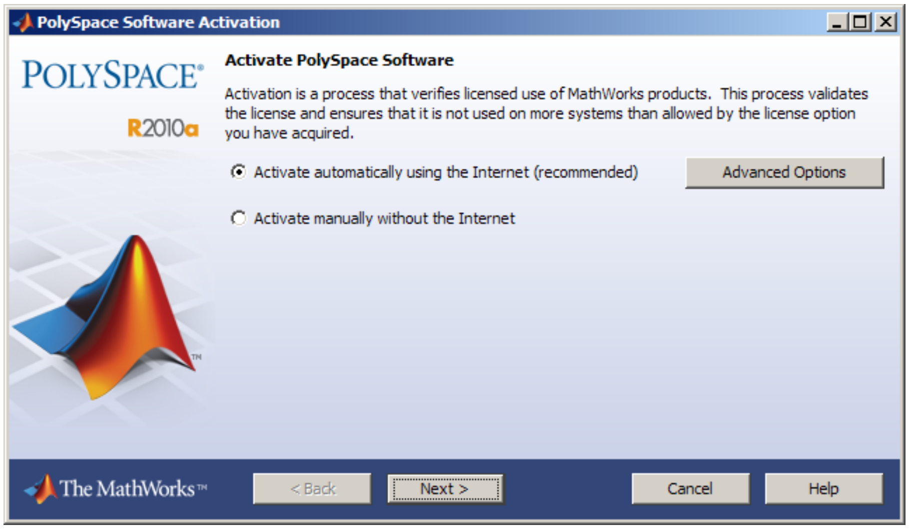
License Activation

PolySpace products now support the MathWorks software activation mechanism.

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

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

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



Activate PolySpace Software

Activation is a process that verifies licensed use of MathWorks products. This process validates the license and ensures that it is not used on more systems than allowed by the license option you have acquired.

[✓ Activate automatically using the Internet (recommended)]

Activate manually without the Internet

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

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

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

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

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

Data Range Specifications for Custom Simulink Data Objects

PolySpace Model Link™ SL software now accepts every Simulink or `mpt` object containing min and max values.

In previous releases, the software did not create DRS entries for custom Simulink® Data Objects, only for `Simulink.Parameter`, `mpt.Parameter`, `Simulink.Signal`, and `mpt.Signal`.

Compatibility Considerations. Verification results may change when compared to previous versions of the software, due to data ranges being applied to additional objects.

Simulink Software Support

Added support for Simulink Version 7.5 (R2010a).

Version 7.1 (R2009b) PolySpace for C/C++ Products

This table summarizes what's new in V7.1 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for C/C++ Product" on page 42
- "PolySpace® Server for C/C++ Product" on page 47

PolySpace Client for C/C++ Product

Report Generator

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

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

- **Coding Rules Report** – Provides information about compliance with MISRA-C Coding Rules, as well as PolySpace configuration settings for the verification.
- **Developer Report** – Provides information useful to developers, including summary results, detailed lists of red, orange, and gray checks, and PolySpace configuration settings for the verification.
- **Developer with Green Checks Report** – Provides the same content as the Developer Report, but also includes a detailed list of green checks.

- **Quality Report** – Provides information useful to quality engineers, including summary results, statistics about the code, graphs showing distributions of checks per file, and PolySpace configuration settings for the verification.

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

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

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

For more information, see "Generating Reports of Verification Results" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Viewer Enhancements

Enhanced Viewer displays results with tooltips containing the values of variables, operands, function parameters, and return values.

You can see range information associated with variables and operators within the source code view.

Note The displayed range information represents a superset of dynamic values, which the software computes using static methods.

If a line of code is all the same color, selecting the line opens an Expanded Source Code window. Place your cursor over the required operator or variable in this window to view range information.

If a line of code contains different colored checks, selecting a check displays the error or warning message along with range information in the selected check view.

For more information, see "Using Range Information in the Viewer" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Global Data Graphs

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

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

For more information, see "Displaying the Access Graph for Variables" in the *PolySpace Products for C User's Guide* or *PolySpace Products for C++ User's Guide*.

Unit-by-unit Verification

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

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

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

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

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

For more information, see "Running Verification Unit-by-Unit" in the *PolySpace Products for C Reference* or *PolySpace Products for C++ Reference*.

Changes to Coding Rules Checker Results

- "Compatibility Considerations" on page 45
- "MISRA-C Rule 5.1 Analysis Improved" on page 45
- "MISRA-C Rule 5.2 Analysis Improved" on page 45
- "MISRA-C Rule 5.7 Analysis Improved" on page 45
- "MISRA-C Rule 8.10 Analysis Improved" on page 46
- "MISRA-C Rule 10.1 Analysis Relaxed" on page 46
- "MISRA-C Rule 10.5 Analysis Improved" on page 46
- "MISRA-C Rule 12.7 Analysis Improved" on page 46
- "MISRA-C Rule 15.0 Analysis Improved" on page 46
- "MISRA-C Rule 16.4 Analysis Improved" on page 46

Compatibility Considerations. Due to changes in the coding rules checker, the number of coding rule violations may change when compared to previous versions of the software.

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

MISRA-C Rule 5.1 Analysis Improved. The coding rules checker now applies MISRA-C rule 5.1 to all identifiers external and internal.

MISRA-C Rule 5.2 Analysis Improved. The coding rules checker now detects violations of MISRA-C Rule 5.2 when the declaration in the outer scope occurs after the declaration in the inner scope.

MISRA-C Rule 5.7 Analysis Improved. The coding rules checker now detects violations of MISRA-C Rule 5.7 in local reused identifiers.

MISRA-C Rule 8.10 Analysis Improved. Only the last declaration takes precedence for `static` or `extern`. The coding rules checker no longer reports violations of MISRA-C Rule 8.10 if the last declaration is static.

MISRA-C Rule 10.1 Analysis Relaxed. The coding rules checker has relaxed enforcement of MISRA-C Rule 10.1 for `x` in `[x]` for any type of expression `x`.

MISRA-C Rule 10.5 Analysis Improved. The coding rules checker now detects violations of MISRA-C Rule 10.5 in expressions with constants.

For example:

```
c = (uint8_t)(ui8 << ( 1U << 2U ) );
```

MISRA-C Rule 12.7 Analysis Improved. The coding rules checker now detects violations of MISRA-C Rule 12.7 in expressions with constants.

For example:

```
~(i=1);
```

MISRA-C Rule 15.0 Analysis Improved. The coding rules checker now detects violations of MISRA-C Rule 15.0 in all statements between switch and first case clause (label, harmless statement).

In addition the coding rules checker now detects jumps and label statements.

MISRA-C Rule 16.4 Analysis Improved. The coding rules checker now keeps the names of the parameters of the first declaration, and reports violations of MISRA-C Rule 16.4 for each occurence.

Operating System Support

Added support for Windows Server® 2008.

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for C/C++ Product

Operating System Support

Added support for Windows Server 2008.

For more information, see the *PolySpace Installation Guide*.

Version 5.4 (R2009b) PolySpace for Ada and Model Link Products

This table summarizes what's new in V5.4 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for Ada Product" on page 48
- "PolySpace® Server for Ada Product" on page 55
- "PolySpace Model Link SL Product" on page 55

PolySpace Client for Ada Product

Report Generator

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

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

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

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

- HTML
- PDF
- RTF
- Microsoft Word
- XML

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

For more information, see "Generating Reports of Verification Results" in the *PolySpace Products for Ada User's Guide* .

Main Generator Enhancements

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

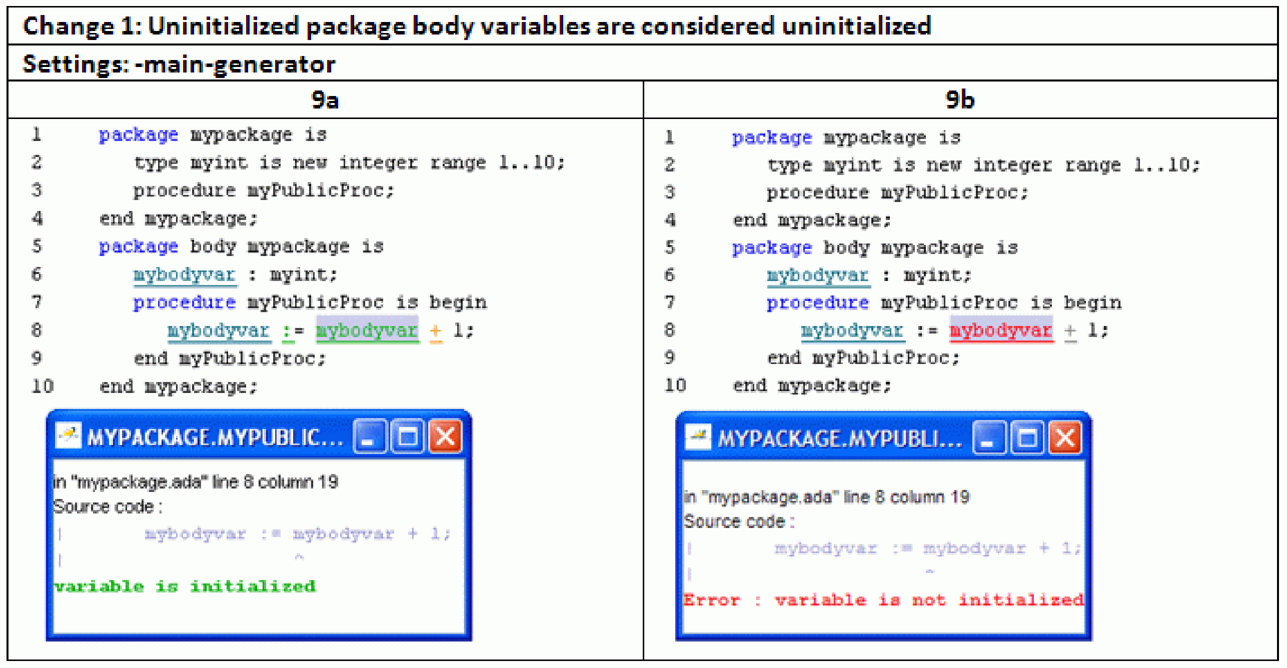
This change may affect your results compared with previous releases, and how you interpret the new results. Specific changes include:

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

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

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

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



Change 1: Uninitialized package body variables are considered uninitialized

Settings: -main-generator

9a

```
1     package mypackage is
2        type myint is new integer range 1..10;
3        procedure myPublicProc;
4     end mypackage;
5     package body mypackage is
6        mybodyvar : myint;
7        procedure myPublicProc is begin
8           mybodyvar := mybodyvar + 1;
9        end myPublicProc;
10    end mypackage;
```

MYPACKAGE.MYPUBLIC...

```
in "mypackage.ada" line 8 column 19
Source code :
|        mybodyvar := mybodyvar + 1;
|                          ^
variable is initialized
```

9b

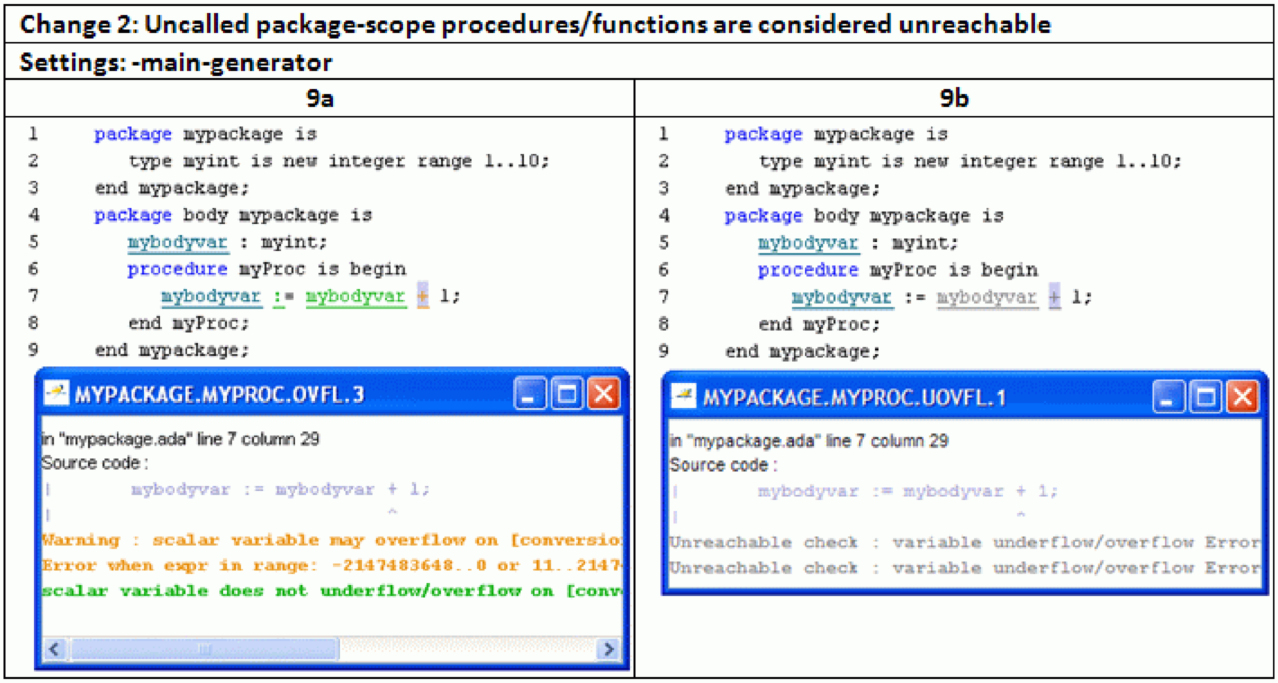
```
1     package mypackage is
2        type myint is new integer range 1..10;
3        procedure myPublicProc;
4     end mypackage;
5     package body mypackage is
6        mybodyvar : myint;
7        procedure myPublicProc is begin
8           mybodyvar := mybodyvar + 1;
9        end myPublicProc;
10    end mypackage;
```

MYPACKAGE.MYPUBLI...

```
in "mypackage.ada" line 8 column 19
Source code :
|        mybodyvar := mybodyvar + 1;
|                          ^
Error : variable is not initialized
```

Uncalled Package-scope Procedures/Functions are Considered Unreachable . Procedures and functions that are declared in the package but not called by code within the package body provided for the verification will now be considered unreachable (gray).

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



Change 2: Uncalled package-scope procedures/functions are considered unreachable

Settings: -main-generator

9a

```
1     package mypackage is
2        type myint is new integer range 1..10;
3     end mypackage;
4     package body mypackage is
5        mybodyvar : myint;
6        procedure myProc is begin
7           mybodyvar := mybodyvar + 1;
8        end myProc;
9     end mypackage;
```

MYPACKAGE.MYPROC.OVFL.3

```
in "mypackage.ada" line 7 column 29
Source code :
|        mybodyvar := mybodyvar + 1;
|                               ^
Warning : scalar variable may overflow on [conversio
Error when expr in range: -2147483648..0 or 11..2147
scalar variable does not underflow/overflow on [conv
```

9b

```
1     package mypackage is
2        type myint is new integer range 1..10;
3     end mypackage;
4     package body mypackage is
5        mybodyvar : myint;
6        procedure myProc is begin
7           mybodyvar := mybodyvar + 1;
8        end myProc;
9     end mypackage;
```

MYPACKAGE.MYPROC.UOVFL.1

```
in "mypackage.ada" line 7 column 29
Source code :
|        mybodyvar := mybodyvar + 1;
|                               ^
Unreachable check : variable underflow/overflow Error
Unreachable check : variable underflow/overflow Error
```

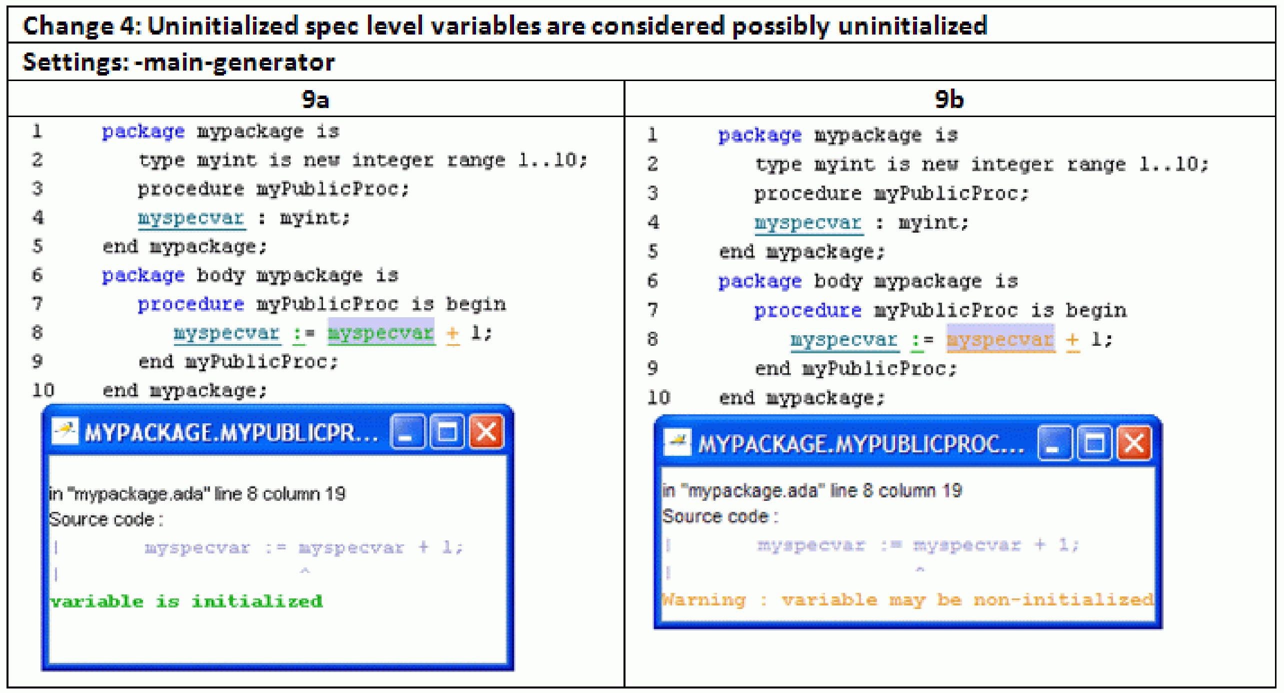
Functions/Procedures Declared at the Spec Level are Called Only Once. Functions or procedures declared at the specification level are called only once.

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

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

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

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



Change 4: Uninitialized spec level variables are considered possibly uninitialized

Settings: -main-generator

9a

```
1     package mypackage is
2        type myint is new integer range 1..10;
3        procedure myPublicProc;
4        myspecvar : myint;
5     end mypackage;
6     package body mypackage is
7        procedure myPublicProc is begin
8           myspecvar := myspecvar + 1;
9        end myPublicProc;
10    end mypackage;
```

MYPACKAGE.MYPUBLICPR...

```
in "mypackage.ada" line 8 column 19
Source code :
|        myspecvar := myspecvar + 1;
|                     ^
variable is initialized
```

9b

```
1     package mypackage is
2        type myint is new integer range 1..10;
3        procedure myPublicProc;
4        myspecvar : myint;
5     end mypackage;
6     package body mypackage is
7        procedure myPublicProc is begin
8           myspecvar := myspecvar + 1;
9        end myPublicProc;
10    end mypackage;
```

MYPACKAGE.MYPUBLICPROC...

```
in "mypackage.ada" line 8 column 19
Source code :
|        myspecvar := myspecvar + 1;
|                     ^
Warning : variable may be non-initialized
```

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

Global Data Graphs

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

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

For more information, see "Displaying the Access Sequence for Variables" in the *PolySpace Products for Ada User's Guide* .

Unit-by-unit Verification

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

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

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

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

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

For more information, see "Running Verification Unit-by-Unit" in the *PolySpace Products for Ada Reference*.

Operating System Support

Added support for Windows Server 2008.

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for Ada Product

Operating System Support

Added support for Windows Server 2008.

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link SL Product

Simulink Software Support

Added support for Simulink Version 7.4 (R2009b).

Version 7.0 (R2009a) PolySpace for C/C++ Products

This table summarizes what's new in V7.0 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	No

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for C/C++ Product" on page 56
- "PolySpace® Server for C/C++ Product" on page 61

PolySpace Client for C/C++ Product

JSF++ Support

Enhanced JSF C++ checker supports all checkable Joint Strike Fighter Air Vehicle C++ coding standards (JSF++:2005).

PolySpace software can now check all possible C++ programming rules defined by Lockheed Martin® for the JSF program. These coding standards are designed to improve the robustness of C++ code, and improve maintainability.

For more information, see "Checking Coding Rules", in the *PolySpace Products for C++ User's Guide*.

Back to Source Link

New "back-to-source" link in the PolySpace launcher associates compile errors, MISRA-C violations, and JSF++ violations reported in the logs directly to the source file.

For more information, see “Examining the MISRA C® Log”in the *PolySpace Products for C User’s Guide* or “Examining the JSF C++ Log”, in the *PolySpace Products for C++ User’s Guide*.

Eclipse Integration

New PolySpace integration with the Eclipse IDE, Version 3.3.

The PolySpace Client for C/C++ product can be integrated with the Eclipse Integrated Development Environment through the PolySpace C/C++ plug-in for Eclipse IDE.

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

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

For more information, see “Using PolySpace Software in the Eclipse IDE” in the *PolySpace Products for C User’s Guide*or
“Using PolySpace Software in the Eclipse IDE”in the *PolySpace Products for C++ User’s Guide*.

Performance Improvements for Multi-Core Systems

Enhanced performance on multi-core architecture platforms, improving the speed of PolySpace code verification.

The time required to perform an average code verification has been reduced. On multi-core systems, you can now select the number of processes that can run simultaneously, further improving performance.

For more information, see “-max-processes” in the *PolySpace Products for C Reference*or *PolySpace Products for C++ Reference*.

Architecture Improvements

Several changes have been made to the PolySpace architecture to improve overall performance, as well as the precision of verification results.

During each verification phase (pass), the software now only analyzes those procedures that need to be analyzed. This means that starting with PASS1, if the verification cannot be more precise than that already completed in a previous pass, the procedure is not analyzed again. This improves the overall performance of the verification. It also means that some passes will finish more quickly than others, and some passes could be completely empty. This is normal behavior.

In addition, these architecture improvements result in the following changes:

- The `quick` precision option is now obsolete, and has been removed. `quick` mode has been replaced with verification PASS0. PASS0 takes somewhat longer to run, but the results are more complete. The limitations of `quick` mode, (no NTL or NTC checks, no float checks, no variable dictionary) no longer apply. Unlike `quick` mode, PASS0 also provides full navigation in the Viewer.
- The `voa` option is now obsolete, and has been removed. Value On Assignment checks are now provided by default. In general, this means that PolySpace results now contain many more VOA checks. For C applications, all possible VOA are given.
- The UOVFL (Float Underflows and Overflows) check no longer exists. Float underflows and overflows are now reported as two separate checks. This is similar to the way integers are handled.

Note Since the single UOVFL check has been replaced by two checks, the total number of checks reported by PolySpace on a given file may be different in this release than with previous versions of the software.

- Messages have been improved for float arithmetic checks, making them similar to the messages for integers. For example, NIV checks on float variables now contain the type size (32 or 64).

- For IPT (Inspection Point) checks, there is now one check for each variable. Previously there was a single IPT check (on the keyword) for multiple variables.
- The log file has several additions, including the names of each PASS, the verification phases, and additional messages.

Compatibility Considerations. The verification results provided by PolySpace software may be different in R2009a than with previous releases of the software. Verification results are more precise, and the total number of checks reported on a given source file may be different. In general, the software now reports more checks, due to increased VOA checks, changes to the IPT check, and the single float UOVFL check being replaced by two checks (UNFL and OVFL).

In addition, due to the float UOVFL check being split into two checks, the selectivity (number of proven checks red+green+gray / number of total checks) of a verification may change significantly for applications using many float variables. For example, an application that had 10 orange UOVFL checks with a previous release, could now have up to 20 orange UNFL and OVFL checks on the same float variables. Although this appears to be a decrease in precision, the verification itself is not less precise.

Mathematical Functions Included in Stubs

Mathematical functions are now included in the standard stubs. This means:

- An IRV (Initialized Return Value) check appears on the math function call.
- The POW check no longer appears in the Viewer.
- Math functions appear in the call graph.
- The modeling of mathematical functions is visible through the stub body, instead of being handled internally.
- By default, math functions are launched with the option `-context-sensitivity` , allowing them to distinguish their calling sites.

In addition, you can provide your own math functions instead of using the standard stub provided by PolySpace software. This allows the software to verify the body of the math function, instead of using a stub for the math function.

For example, in C90, the mathematical function `fabs()` has the prototype:

```
double fabs(double) ;
```

However, on a 16-bit target, the function may have the prototype:

```
float fabs(float);
```

In this case, you would want to verify your own `fabs()` function.

To provide your own math function:

1. Create source code for the function. For example:

   ```
   float  fabs (float var)
   {
     if (var >= 0.0f)
       return var;
     return -var;
   }
   ```

2. Provide the function to your verification using the `D` compiler flag. For example:

   ```
   polyspace-c -D  __polyspace_no_fabs
   ```

Note There is a compiler flag for each standard ANSI C90 mathematical function. A complete list of flags is located in the file: `%POLYSPACE_C%\Verifier\cinclude\__polyspace__stdstubs.c`.

Compatibility Considerations. Since the POW check no longer appears in the Viewer, verification results may be different in R2009a than with previous releases of the software.

Character Encoding Options

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

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

For more information, see "Setting Character Encoding Preferences"in the *PolySpace Products for C User's Guide*or *PolySpace Products for C++ User's Guide*.

Automatic Orange Tester

The Automatic Orange Tester (for C), dynamically stresses unproven code (orange checks) to help you identify run-time errors.

For more information, see "Automatically Testing Orange Code" in the *PolySpace Products for C User's Guide*.

Compatibility Considerations. If you open verification results created with an older version of the product in the Automatic Orange Tester, you may get a compilation error. The version of the product used to create the instrumented source code must be the same as the one used for analysis in the Automatic Orange Tester.

To avoid this problem, re-launch the code verification with the current version of the product.

Operating System Support

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for C/C++ Product

Performance Improvements for Multi-Core Systems

Enhanced performance on multi-core architecture platforms, improving the speed of PolySpace code verification.

The time required to perform an average code verification has been reduced. On multi-core systems, you can now select the number of processes that can run simultaneously, further improving performance.

For more information, see “-max-processes” in the *PolySpace Products for C Reference*or *PolySpace Products for C++ Reference*.

Architecture Improvements

Several changes have been made to the PolySpace architecture to improve overall performance, as well as the precision of verification results.

During each verification phase (pass), the software now only analyzes those procedures that need to be analyzed. This means that starting with PASS1, if the verification cannot be more precise than that already completed in a previous pass, the procedure is not analyzed again. This improves the overall performance of the verification. It also means that some passes will finish more quickly than others, and some passes could be completely empty. This is normal behavior.

In addition, these architecture improvements result in the following changes:

- The `quick` precision option is now obsolete, and has been removed. `quick` mode has been replaced with verification PASS0. PASS0 takes somewhat longer to run, but the results are more complete. The limitations of `quick` mode, (no NTL or NTC checks, no float checks, no variable dictionary) no longer apply. Unlike `quick` mode, PASS0 also provides full navigation in the Viewer.
- The `voa` option is now obsolete, and has been removed. Value On Assignment checks are now provided by default. In C, all possible VOA are given.
- The UOVFL (Float Underflows and Overflows) check no longer exists. Float underflows and overflows are now reported as two separate checks. This is similar to the way integers are handled.

Note Since the single UOVFL check has been replaced by two checks, the total number of checks reported by PolySpace on a given file may be different in this release than with previous versions of the software.

- Messages have been improved for float arithmetic checks, making them similar to the messages for integers. For example, NIV checks on float variables now contain the type size (32 or 64).
- For IPT (Inspection Point) checks, there is now one check for each variable. Previously there was a single IPT check (on the keyword) for multiple variables.
- The log file has several additions, including the names of each PASS, the verification phases, and additional messages.

Compatibility Considerations. The verification results provided by PolySpace software may be different in R2009a than with previous releases of the software. Verification results are more precise, and the total number of checks reported on a given source file may be different. In general, the software now reports more checks, due to increased VOA checks, changes to the IPT check, and the single float UOVFL check being replaced by two checks (UNFL and OVFL).

In addition, due to the float UOVFL check being split into two checks, the selectivity (number of proven checks red+green+gray / number of total checks) of a verification may change significantly for applications using many float variables. For example, an application that had 10 orange UOVFL checks with a previous release, could now have up to 20 orange UNFL and OVFL checks on the same float variables. Although this appears to be a decrease in precision, the verification itself is not less precise.

Operating System Support

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

For more information, see the *PolySpace Installation Guide*.

Version 5.3 (R2009a) PolySpace for Ada and Model Link Products

This table summarizes what's new in V5.3 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports PolySpace Model Link TL Bug Reports PolySpace UML Link RH Bug Reports	No

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for Ada Product" on page 64
- "PolySpace® Server for Ada Product" on page 65
- "PolySpace Model Link SL Product" on page 65
- "PolySpace UML Link RH Product" on page 66

PolySpace Client for Ada Product

Character Encoding Options

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

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

For more information, see “Setting Character Encoding Preferences” in the *PolySpace Products for Ada User’s Guide*.

Operating System Support

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for Ada Product

Operating System Support

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link SL Product

PolySpace Menu Option in Simulink

New option in the Simulink Tools menu to launch PolySpace software directly from Simulink.

For more information, see “Starting the PolySpace Verification”in the *PolySpace Model Link Products User’s Guide*.

Manual Selection of Data Range Specifications (DRS) File

You can now manually select a Data Range Specification (DRS) file within Simulink, instead of accepting the default DRS file.

For more information, see “Data Range Specification”in the *PolySpace Model Link Products User’s Guide*.

Simulink Software Support

Added support for Simulink Version 7.3 (R2009a).

PolySpace UML Link RH Product

Rhapsody Support

Added support for Telelogic® Rhapsody® Version 7.2 and 7.3.

Version 6.0 (R2008b) PolySpace for C/C++ Products

This table summarizes what's new in V6.0 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports	No.

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for C/C++ Product" on page 67
- "PolySpace® Server for C/C++ Product" on page 68

PolySpace Client for C/C++ Product

Automatic Orange Tester

Automatic Orange Tester (for C), dynamically stresses unproven code (orange checks) to identify run-time errors, and provides information to help you identify the cause of these errors.

The Automatic Orange Tester complements the results review in the Viewer module of PolySpace Client for C/C++ by automatically creating test cases for all input variables in orange code, and then dynamically testing the code to find actual runtime errors. The Automatic Orange Tester also provides detailed information on why each test-case failed. You can use this information to quickly identify the cause of the error, and determine if there is an actual bug in the code.

For more information, see "Automatically Testing Orange Code" in the *PolySpace Products for C User's Guide*.

JSF++ Support

Support for a subset of the Joint Strike Fighter Air Vehicle C++ coding standards (JSF++:2005).

PolySpace software can now check 120 of the C++ programming rules defined by Lockheed Martin for the JSF program. These coding standards are designed to improve the robustness of C++ code, and improve maintainability.

For more information, see “Checking Coding Rules”, in the *PolySpace Products for C++ User’s Guide*.

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for C/C++ Product

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

Version 5.2 (R2008b) PolySpace for Ada and Model Link Products

This table summarizes what's new in V5.2 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Includes fixes: PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports	No.

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for Ada Product" on page 69
- "PolySpace® Server for Ada Product" on page 70
- "PolySpace Model Link SL Product" on page 70
- "PolySpace Model Link TL Product" on page 71
- "PolySpace UML Link RH Product" on page 71

PolySpace Client for Ada Product

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for Ada Product

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link SL Product

Model Reference Support

Added support for Simulink Model Reference.

PolySpace Model Link SL software now automatically detects model references in Simulink models, allowing you to quickly track any verification issues back to the original model.

For more information, see the *PolySpace Model Link Products User's Guide*.

Stateflow Chart Support

Added support for Stateflow® Charts within Simulink models.

PolySpace Model Link SL software now supports Stateflow Charts within Simulink models, allowing you to quickly track any verification issues back to the original Stateflow chart. In addition, any Stateflow comments are now highlighted in the PolySpace source code view.

For more information, see the *PolySpace Model Link Products User's Guide*.

Simulink Software Support

Added support for Simulink Version 7.2 (R2008b).

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link TL Product

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

PolySpace UML Link RH Product

Ada Language Support

Added support for Ada language in Rhapsody software.

For more information, see the *PolySpace UML Link™ RH User's Guide*.

Operating System Support

Added support for 64–bit Linux.

For more information, see the *PolySpace Installation Guide*.

Version 5.1 (R2008a) PolySpace Software

This table summarizes what's new in V5.1 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as **Compatibility Considerations**, below. See also Summary.	Includes fixes: PolySpace Client for C/C++ Bug Reports PolySpace Server for C/C++ Bug Reports PolySpace Client for Ada Bug Reports PolySpace Server for Ada Bug Reports PolySpace Model Link SL Bug Reports	No.

New features and changes introduced in this version are organized by product:

- "PolySpace® Client for Ada Product" on page 73
- "PolySpace® Server for Ada Product" on page 75
- "PolySpace® Client for C/C++ Product" on page 76
- "PolySpace® Server for C/C++ Product" on page 79
- "PolySpace Model Link SL Product" on page 81
- "PolySpace Model Link TL Product" on page 81
- "PolySpace UML Link RH Product" on page 82

PolySpace Client for Ada Product

Removed Cygwin Software Dependency for Windows Platforms

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

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

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

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

Commands	Previous Location	New Location
Standard	`*PolySpaceInstallDir*\verifier\bin\`	`*PolySpaceInstallDir*\verifier\wbin\`
Remote Launcher	`*PolySpace_Common*\RemoteLauncher\bin\`	`*PolySpace_Common*\RemoteLauncher\wbin\`
Viewer	`*PolySpace_Common*\Viewer\bin\`	`*PolySpace_Common*\Viewer\wbin\`

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

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

```
PolySpaceInstallDir\verifier\tools\perl\win32\bin\perl.exe
```

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Viewer Improvements

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

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

Enhanced Compilation Checks

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

One-Click Enhancements

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

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for Ada Product

Removed Cygwin Software Dependency for Windows Platforms

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

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

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

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

Commands	Previous Location	New Location
Standard	`*PolySpaceInstallDir*\verifier\bin\`	`*PolySpaceInstallDir*\verifier\wbin\`
Remote Launcher	`*PolySpace_Common*\RemoteLauncher\bin\`	`*PolySpace_Common*\RemoteLauncher\wbin\`
Viewer	`*PolySpace_Common*\Viewer\bin\`	`*PolySpace_Common*\Viewer\wbin\`

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

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

```
PolySpaceInstallDir\verifier\tools\perl\win32\bin\perl.exe
```

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Client for C/C++ Product

Removed Cygwin Software Dependency for Windows Platforms

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

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

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

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

Commands	Previous Location	New Location
Standard	`PolySpaceInstallDir\verifier\bin\`	`PolySpaceInstallDir\verifier\wbin\`
Remote Launcher	`PolySpace_Common\RemoteLauncher\bin\`	`PolySpace_Common\RemoteLauncher\wbin\`
Viewer	`PolySpace_Common\Viewer\bin\`	`PolySpace_Common\Viewer\wbin\`

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

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

```
PolySpaceInstallDir\verifier\tools\perl\win32\bin\perl.exe
```

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Viewer Improvements

Enhanced exploring capability in the viewer to provide more precise locations for C++ checks.

The source code view of the PolySpace viewer now displays the location of C++ checks more accurately.

One-Click Enhancements

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

For more information, see "Day to Day Use " in the *PolySpace Products for C User's Guide*.

Generic Target Option for C++

New Generic Target option for C++, to allow custom target processors. The Generic Target option for C++ is similar to the previous Generic Target for C.

For more information, see "Setting Up Project for Generic Target Processors" in the *PolySpace Products for C++ User's Guide*.

Class Analyzer Enhancements for C++

Enhanced class analyzer now calls all private constructors and destructors.

Previously, the sources analyzed were generally non-inherited public or protected methods of the class. In version 5.1, the functions that are analyzed include all non-inherited constructors and destructors, and all non-inherited public or protected methods of the class.

For more information, see "PolySpace Class Analyzer" in the *PolySpace Products for C++ User's Guide*.

GNU Compiler Support for C++

New support for the GNU® compiler (GCC 3.4) for C++.

The new GNU dialect option supports variable length arrays, anonymous structures, and other constructions allowed by GCC.

For more information, see "Dialect Issues" in the *PolySpace Products for C++ User's Guide*.

PolySpace C++ Add-in for Visual Studio

Simplified user interface for PolySpace C++ add-in for Microsoft® Visual Studio®.

The PolySpace Browser tab has been eliminated from the Visual Studio® window. To perform an analysis of a file in Visual Studio, you now simply right-click on the file and select **Start PolySpace**.

For more information, see "Using PolySpace Software in Visual Studio" in the *PolySpace Products for C++ User's Guide*.

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Server for C/C++ Product

Removed Cygwin Software Dependency for Windows Platforms

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

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

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

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

Commands	Previous Location	New Location
Standard	`*PolySpaceInstallDir*\verifier\bin\`	`*PolySpaceInstallDir*\verifier\wbin\`
Remote Launcher	*PolySpace_Common*\ `RemoteLauncher\bin\`	*PolySpace_Common*\ `RemoteLauncher\wbin\`
Viewer	*PolySpace_Common*\ `Viewer\bin\`	*PolySpace_Common*\ `Viewer\wbin\`

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

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

```
PolySpaceInstallDir\verifier\tools\perl\win32\bin\perl.exe
```

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

GNU Compiler Support for C++

New support for the GNU compiler (GCC 3.4) for C++.

The new GNU dialect option supports variable length arrays, anonymous structures, and other constructions allowed by GCC.

For more information, see "Dialect Issues" in the *PolySpace Products for C++ User's Guide*.

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link SL Product

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Simulink Software Support

Added support for Simulink Version 7.1 (R2008a).

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace Model Link TL Product

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

PolySpace UML Link RH Product

Enhanced Installer

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

For more information, see the *PolySpace Installation Guide*.

Rhapsody Support

Added support for Telelogic Rhapsody Version 7.1.

C Language Support

Added support for C language in Rhapsody software.

For more information, see the *PolySpace UML Link RH User's Guide*.

Operating System Support

Added support for the following operating systems:

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

For more information, see the *PolySpace Installation Guide*.

Compatibility Summary for PolySpace Software

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided in the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version for C/C++ V7.2 (R2010a)	See the **Compatibility Considerations** subheading for these new features or changes: • "Importing Review Comments" on page 10 • "Data Range Specifications (DRS) Enhancements" on page 11 • "Methodological Assistant Enhancements" on page 18 • "Class Analyzer Enhancements for C++" on page 19 • "Change to Time Format in Log File" on page 19 • "Merging of `OVFL` and `UNFL` Checks" on page 19 • "Improved `UNR` Checks" on page 20 • "Changes to Verification Results" on page 21 • "Changes to Coding Rules Checker Results" on page 29

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version for Ada and Model Link Products V5.5 (R2010a)	See the **Compatibility Considerations** subheading for these new features or changes: • "Data Range Specifications for Custom Simulink Data Objects" on page 41
V7.1 for C/C++ (R2009b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Changes to Coding Rules Checker Results" on page 45
V5.4 for Ada (R2009b)	See the **Compatibility Considerations** subheading for this new feature or change: • "Main Generator Enhancements" on page 49
V7.0 for C/C++ (R2009a)	See the **Compatibility Considerations** subheading for these new features or changes: • "Architecture Improvements" on page 58 • "Mathematical Functions Included in Stubs" on page 59 • "Automatic Orange Tester" on page 61
V5.3 for Ada (R2009a)	None
V6.0 for C/C++ (R2008b)	None

Version (Release)	New Features and Changes with Version Compatibility Impact
V5.2 for Ada (R2008b)	None
V5.1 (R2008a)	See the **Compatibility Considerations** subheading for this new feature or change: • "Removed Cygwin Software Dependency for Windows Platforms" on page 73