PolySpace
TECHNOLOGIES

PolySpace™ for Ada Getting Started R2007a+

How to Contact The MathWorks

www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup
www.mathworks.com/contact_TS.html	Technical Support
suggest@mathworks.com	Product enhancement suggestions
bugs@mathworks.com	Bug reports
doc@mathworks.com	Documentation error reports
service@mathworks.com	Order status, license renewals, passcodes
info@mathworks.com	Sales, pricing, and general information

508-647-7000 (Phone)
508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098
For contact information about worldwide offices, see the MathWorks Web site.

Table of Contents

Typographical conventions:

➪ The "➤" symbol indicates an action which must be performed by the user.

➪ `<PolySpaceInstallDir>` stands for the directory/folder name where the PolySpace products were installed.

➪ The "`Courier New`" font is used for mentioning data seen on the screen of the computer.

1. General Requirements

1.1. Computer Configuration

Please refer to PolySpace installation manual for the minimum hardware.
The timing is the following:

- The installation of PolySpace products takes around 5 minutes (see the complete installation guide as available from the PolySpace installation CD-ROM in *\Docs\Install\PolySpace_Install_Guide.pdf*).
- The first step of this tutorial takes about 15 minutes.
- The second step of this tutorial takes about 15 minutes.

1.2. Structure of this document

Once the installation is done, you can launch PolySpace by using the following icon that was placed on your desktop:

This Getting Started will focus on the following three exercises using PolySpace Client, PolySpace Viewer and PolySpace Remote Launcher:

- In Step 1, we will analyze a simple package "`example`" by using PolySpace Client
- In Step 2, we will review the results obtained during Step 1 by using PolySpace Viewer
- In Step 3, we will send an analysis remotely to a server.

2. Step 1: PolySpace Client - Setting up and launching an analysis of a single Ada package

This paragraph describes a basic package analysis. It focuses on the analysis of the "`example`" package, which is included in the PolySpace installation directory and located at:
`<PolySpaceInstallDir>\Examples\Demo_Ada\sources\example.adb.`

The PolySpace analysis process is composed of three main phases:

1. First, PolySpace checks the syntax and semantic of the analyzed file(s). However, as PolySpace is not associated to a particular compiler, **benefits** of this phase are triple for the analysed source code: **Ada Standard compliance, portability** and **maintainability.**
2. Then, PolySpace seeks the main procedure. If none is found, PolySpace Client will generate one automatically. This function will call all the functions which are declared in the specification of the package.
3. Finally, PolySpace proceeds with the code analysis phase, during which run time errors are detected and highlighted in the code.

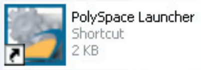
2.1. Analysis prerequisites

Any analysis requires the following:

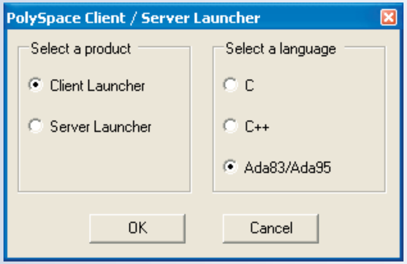
- PolySpace products and their related license file correctly installed;
- Source code files (in this case "`example.adb`") and all others specifications that it may directly or indirectly requires.

2.2. Setting up a PolySpace Client analysis

➤ Double-click on the PolySpace Launcher icon:



A window appears proposing to choose the product to be used for the analysis and the language of the file to be analyzed:



PolySpace Client / Server Launcher

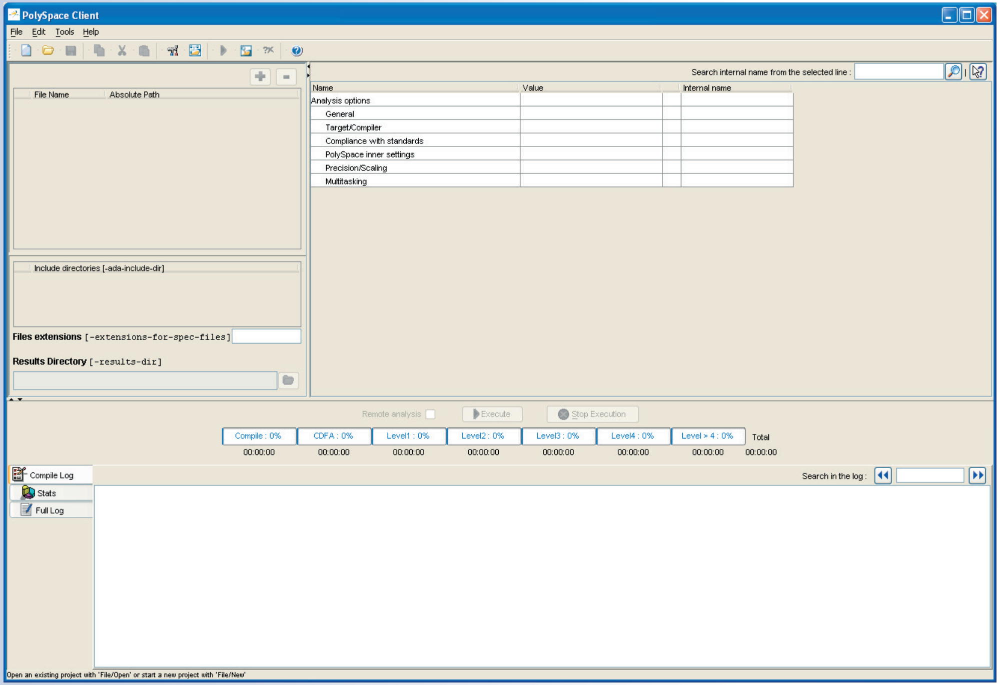
Select a product: [✓ Client Launcher] Server Launcher

Select a language: C C++ [✓ Ada83/Ada95]

If PolySpace is not installed for some languages, these choices of languages will be grayed out.

Step 1: PolySpace Client - Setting up and launching an analysis of a single Ada package

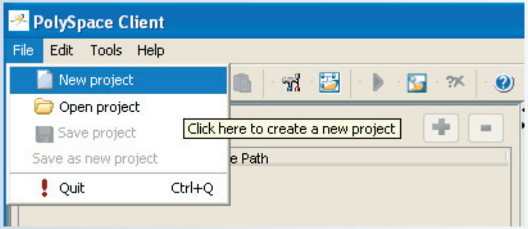
➤ Select "Client Launcher", language "Ada83/Ada95" and then, click on OK.
The Graphical Interface of PolySpace analysis Launcher is displayed as below:



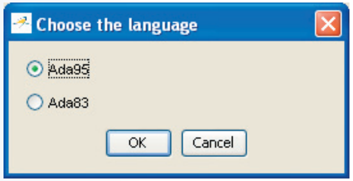
Name	Value	Internal name
Analysis options		
General		
Target/Compiler		
Compliance with standards		
PolySpace inner settings		
Precision/Scaling		
Multitasking		

Open an existing project with 'File/Open' or start a new project with 'File/New'

➤ Click on File/New Project to start an analysis:

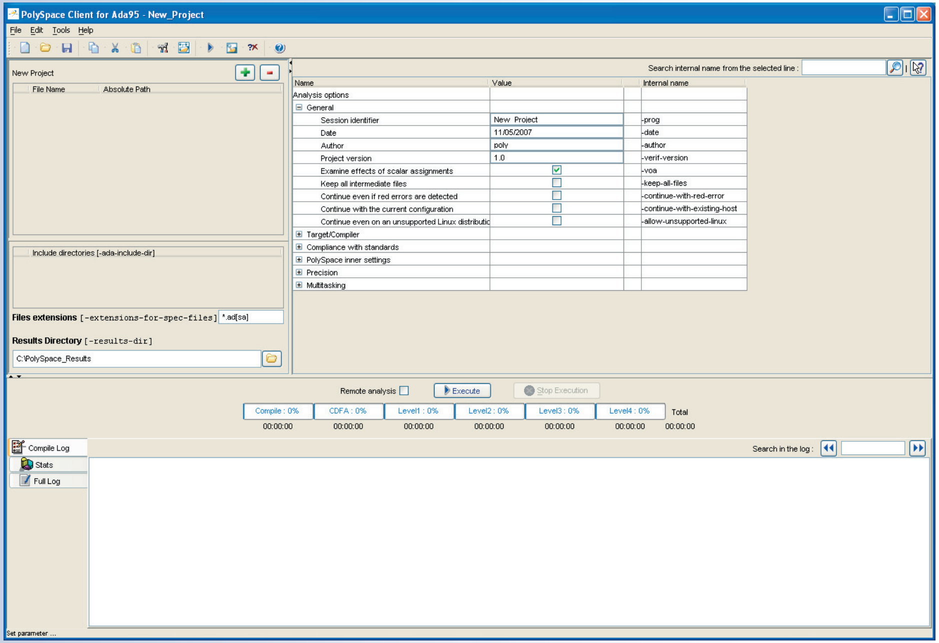


➤ Select Ada95 as the language and click on OK :



The PolySpace Client New Project window opens. It contains four sections:
• At the very top, the title bar, which contains usual icons and menus;
• Top left is the list of files to analyze, along with include and results directories;
• Top right is the set of options associated with the analysis that will be processed;
• The bottom area allows following the execution and progress of the analysis.

Step 1: PolySpace Client - Setting up and launching an analysis of a single Ada package



Name	Value	Internal name
Analysis options		
⊟ General		
Session identifier	New_Project	-prog
Date	11/05/2007	-date
Author	poly	-author
Project version	1.0	-verif-version
Examine effects of scalar assignments	✓	-voa
Keep all intermediate files	☐	-keep-all-files
Continue even if red errors are detected	☐	-continue-with-red-error
Continue with the current configuration	☐	-continue-with-existing-host
Continue even on an unsupported Linux distributic	☐	-allow-unsupported-linux
⊞ Target/Compiler		
⊞ Compliance with standards		
⊞ PolySpace inner settings		
⊞ Precision		
⊞ Multitasking		

Include directories [-ada-include-dir]

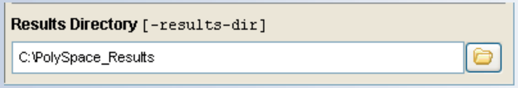
Files extensions [-extensions-for-spec-files] *.ad[sa]

Results Directory [-results-dir]

C:\PolySpace_Results

Compile : 0%	CDFA : 0%	Level1 : 0%	Level2 : 0%	Level3 : 0%	Level4 : 0%	Total
00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	00:00:00

➤ Start by updating the result directory name by clicking on the browse button :

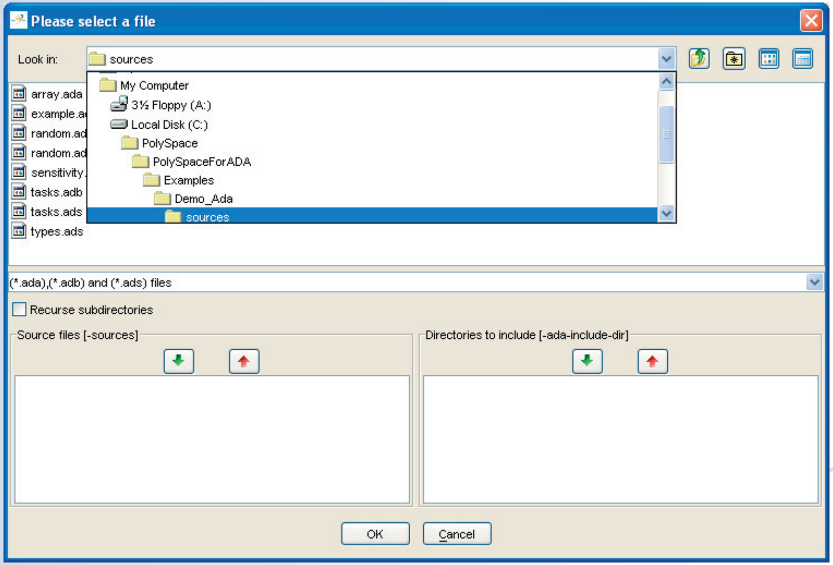


Results Directory [-results-dir]

C:\PolySpace_Results

This directory is the one where PolySpace Client will store the results of the analysis. In this Getting Started, we will choose the default directory: "`C:\PolySpace_Results`"

➤ Now, click on the ✚ button (right of the "`New Project`" label).
It opens the "`Please select a file`" window, from which you can select one or several files to analyse.


Please select a file

Look in: sources

- array.ada
- example.a
- random.ad
- random.ad
- sensitivity.
- tasks.adb
- tasks.ads
- types.ads

- My Computer
- 3½ Floppy (A:)
- Local Disk (C:)
- PolySpace
- PolySpaceForADA
- Examples
- Demo_Ada
- sources

(*.ada),(*.adb) and (*.ads) files

☐ Recurse subdirectories

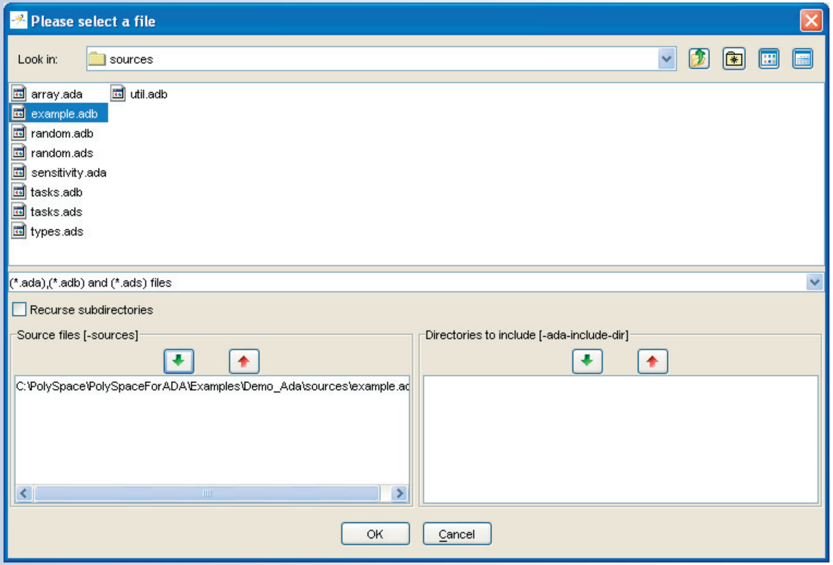
Source files [-sources]

Directories to include [-ada-include-dir]

OK Cancel

➤ In the "`Look in`" section, click on ⌄ and select "`<PolySpaceInstallDir>\Examples\Demo_Ada\sources`". A list of files appears in the box (`<PolySpaceInstallDir>` corresponds to `C:\PolySpace\PolySpaceForAda` in the figure above).

➤ Select "`example.adb`" and click on [⬇] in the "`Source files [-sources]`" section (bottom left) of the window. The file is now listed among the source files to be analyzed.

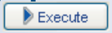


```
C:\PolySpace\PolySpaceForADA\Examples\Demo_Ada\sources\example.ac
```

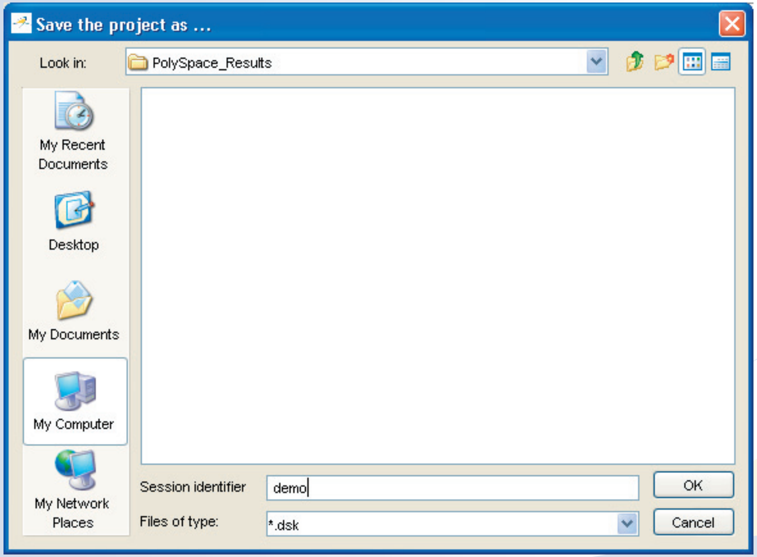
➤ Click on OK to go back to the "`PolySpace Client for Ada95 - New_Project`" window.

Note: it is also possible to drag a directory or source files and drop it in the "`File Name/Absolute Path`" part (top left of PolySpace Client) without using the "`Please select a file`" window.

2.3. PolySpace Client: running the analysis

➤ Click on  to start the analysis. Alternatively, you can click on the button in the title bar to run PolySpace Client with the current setting.

The window titled "`Save the project as`" opens. You can decide where to store the configuration information related to the analysis. Here, create a file called "`demo`" and save it in the PolySpace result directory. The full name of that file will be "`demo.dsk`".



Save the project as ...

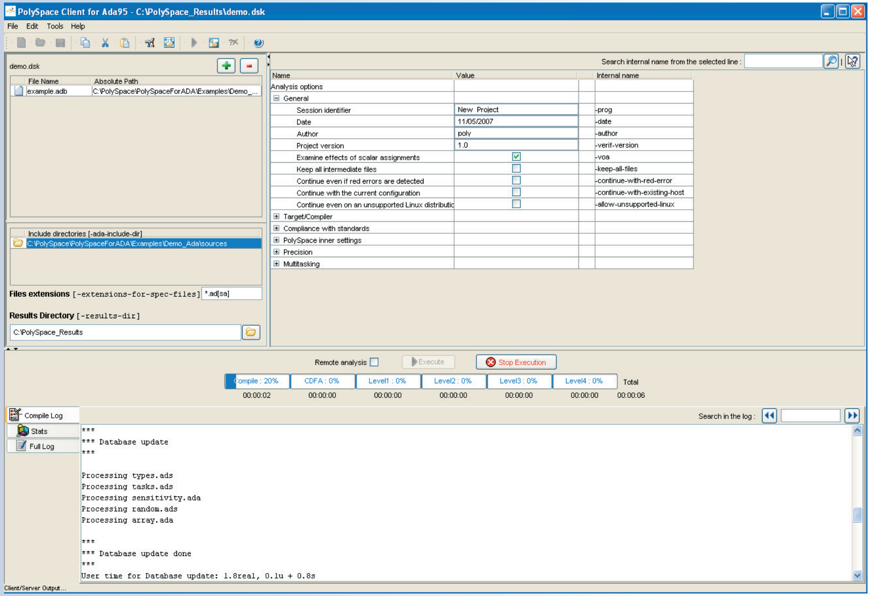
Look in: PolySpace_Results

Session identifier: demo

Files of type: *.dsk

Step 1: PolySpace Client - Setting up and launching an analysis of a single Ada package

➤ Click on OK to go back to the "`PolySpace Client for Ada95 - New_Project`" window and click again on Execute to proceed forward.



demo.dsk

File Name	Absolute Path
example.adb	C:\PolySpace\PolySpaceForADA\Examples\Demo_...

Include directories [-ada-include-dir]
C:\PolySpace\PolySpaceForADA\Examples\Demo_Ada\sources

Files extensions [`-extensions-for-spec-files`] *.ad[sa]

Results Directory [`-results-dir`]
C:\PolySpace_Results

Search internal name from the selected line :

Name	Value	Internal name
Analysis options		
⊟ General		
Session identifier	New Project	-prog
Date	11/05/2007	-date
Author	poly	-author
Project version	1.0	-verif-version
Examine effects of scalar assignments	☑	-voa
Keep all intermediate files	☐	-keep-all-files
Continue even if red errors are detected	☐	-continue-with-red-error
Continue with the current configuration	☐	-continue-with-existing-host
Continue even on an unsupported Linux distributic	☐	-allow-unsupported-linux
⊞ Target/Compiler		
⊞ Compliance with standards		
⊞ PolySpace inner settings		
⊞ Precision		
⊞ Multitasking		

Remote analysis ☐ Execute Stop Execution

Compile : 20%	CDFA : 0%	Level1 : 0%	Level2 : 0%	Level3 : 0%	Level4 : 0%	Total
00:00:02	00:00:00	00:00:00	00:00:00	00:00:00	00:00:00	00:00:06

```
***
*** Database update
***

Processing types.ads
Processing tasks.ads
Processing sensitivity.ada
Processing random.ads
Processing array.ada

***
*** Database update done
***
User time for Database update: 1.8real, 0.1u + 0.8s
```

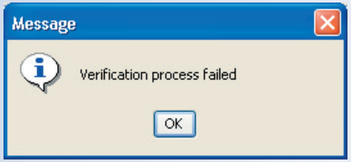
Client/Server Output...

A progress report is displayed in the bottom part of the graphical interface, indicating that the analysis is being performed. The Execute button is also grayed out.

Note: you may use the Stop Execution button - Stop Execution - to interrupt the analysis but it is not part of the current tutorial.

2.3.1. Parsing errors during preliminary PolySpace analysis stages

After some checks, PolySpace will show an error message:

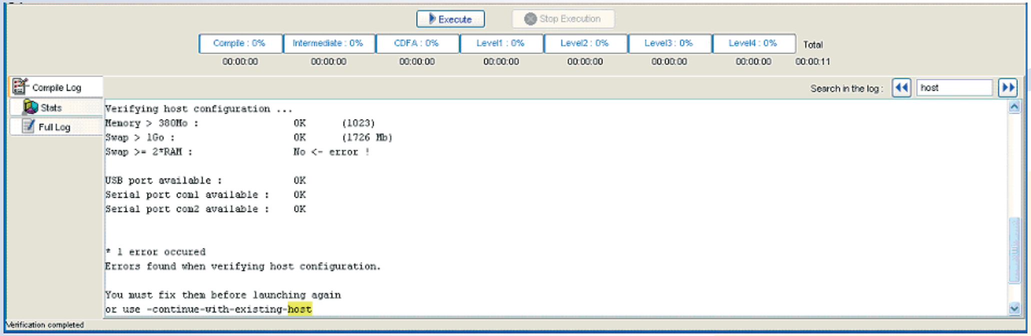


Verification process failed

Let's try and understand why we get this error message.

First possible cause for the error message: Hardware recommendation

If this happens, please verify whether your computer meets the minimal hardware requirements.
A message similar to the one below would be displayed in the bottom part of the graphical interface:



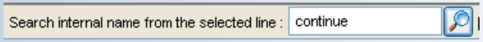
```
Verifying host configuration ...
Memory > 380Mo :              OK       (1023)
Swap > 1Go :                  OK       (1726 Mb)
Swap >= 2*RAM :               No <- error !

USB port available :          OK
Serial port com1 available :  OK
Serial port com2 available :  OK


* 1 error occured
Errors found when verifying host configuration.

You must fix them before launching again
or use -continue-with-existing-host
```

➤ To help you understand the issue, you can search into the log file. Type "`host`" in the "`Search in the log:`" box and click on ◀◀ to check whether the error corresponds to a hardware recommendation problem.

If you have a problem related to host configuration, in order to continue analysis, you can either:

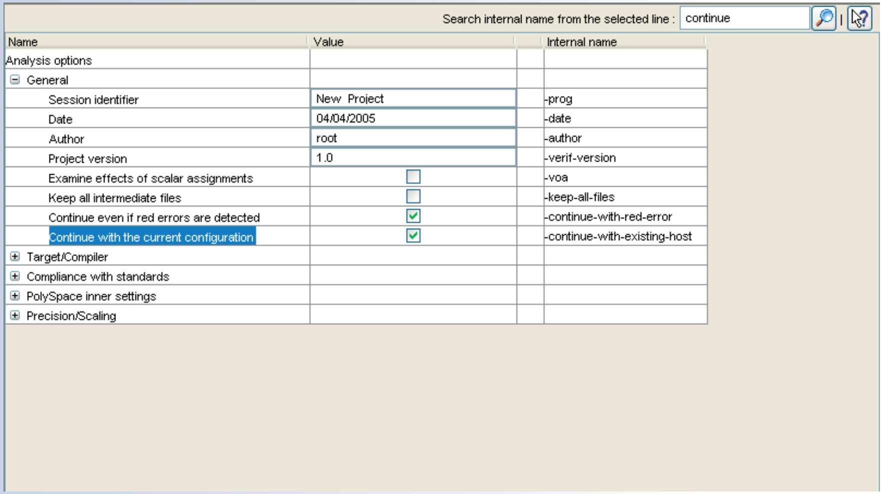
- upgrade your computer to meet the minimal requirements, or
- use the `-continue-with-existing-host` option which overrides the initial check for minimal hardware configuration.

➤ To set up the `-continue-with-existing-host` option, please type "`continue`" in the "`Search internal name from the selected line`" box at the top right of the window



Search internal name from the selected line : continue

➤ Then click on [🔍]. It will show all options containing the word "`continue`" as shown below:

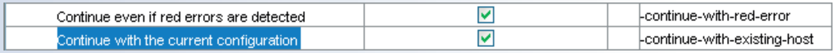


Search internal name from the selected line : continue

Name	Value	Internal name
Analysis options		
⊟ General		
Session identifier	New Project	-prog
Date	04/04/2005	-date
Author	root	-author
Project version	1.0	-verif-version
Examine effects of scalar assignments	☐	-voa
Keep all intermediate files	☐	-keep-all-files
Continue even if red errors are detected	✓	-continue-with-red-error
Continue with the current configuration	✓	-continue-with-existing-host
⊞ Target/Compiler		
⊞ Compliance with standards		
⊞ PolySpace inner settings		
⊞ Precision/Scaling		

- Check the box ✓ in the "Value" column that is associated to the "-continue-with-existing-host" line as shown below.

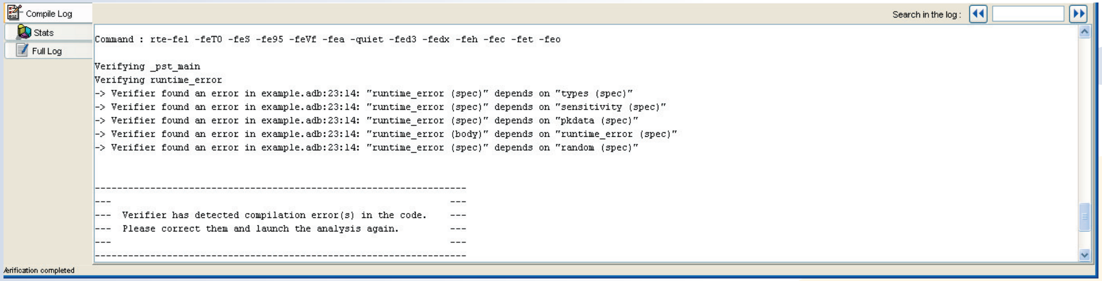
- It is also recommended to select the -continue-with-red-error option. Indeed, "example.adb" contains - on purpose - code with some definite errors, later called red errors. This option allows you to continue the analysis even if red errors are detected. Otherwise, the analysis would just stop after the detection of the first of these errors.



Continue even if red errors are detected	✓	-continue-with-red-error
Continue with the current configuration	✓	-continue-with-existing-host

Second possible cause for the error message: Information about Header files

Another cause of error may be that PolySpace Client misses some package specifications.



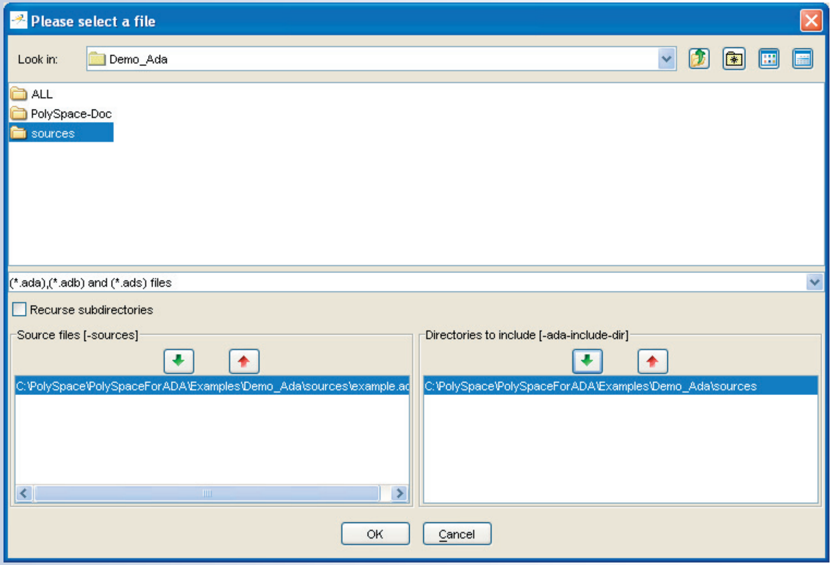
```
Command : rte-fel -feT0 -feS -fe95 -feVf -fea -quiet -fed3 -fedx -feh -fec -fet -feo

Verifying _pst_main
Verifying runtime_error
-> Verifier found an error in example.adb:23:14: "runtime_error (spec)" depends on "types (spec)"
-> Verifier found an error in example.adb:23:14: "runtime_error (spec)" depends on "sensitivity (spec)"
-> Verifier found an error in example.adb:23:14: "runtime_error (spec)" depends on "pkdata (spec)"
-> Verifier found an error in example.adb:23:14: "runtime_error (body)" depends on "runtime_error (spec)"
-> Verifier found an error in example.adb:23:14: "runtime_error (spec)" depends on "random (spec)"


-------------------------------------------------------------------
---                                                             ---
---  Verifier has detected compilation error(s) in the code.    ---
---  Please correct them and launch the analysis again.         ---
---                                                             ---
-------------------------------------------------------------------
```

Verification completed

In the tutorial, as shown above, some specification are missing: "types", "sensitivity", "pkdata", "runtime_error" and "random". To fix theses compilation errors, you need to indicate where to find these specifications. As PolySpace is not associated with one particular compiler, it is mandatory to indicate where library files are stored.

In our "`example.adb`" file analysis, the related specifications are located in the same directory as the `adb` file: `<PolySpaceInstallDir>\Examples\Demo_Ada\sources`.

➤ Open the "`Please select a file`" window by using the ✚ button (right of the "`demo.dsk`" label in the top right of the interface):



C:\PolySpace\PolySpaceForADA\Examples\Demo_Ada\sources\example.ad

C:\PolySpace\PolySpaceForADA\Examples\Demo_Ada\sources

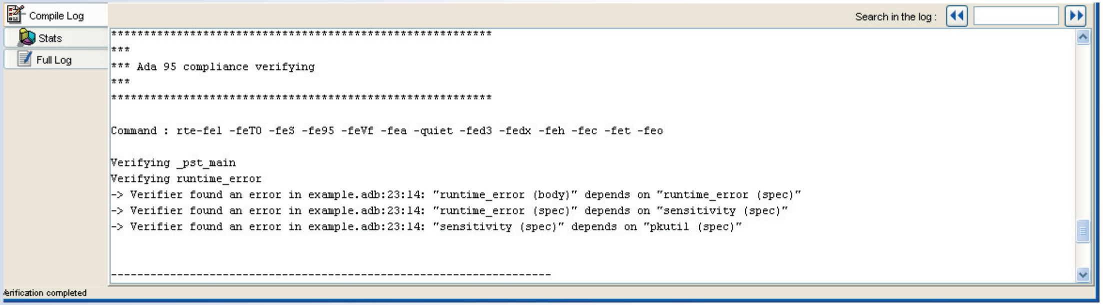
➤ Select "`<PolySpace For Ada>\Examples\Demo_Ada\sources`", where the specifications are located.

➤ Click on ⬇ in the "`Directories to include [-ada-include-dir]`" section, and then close the window using OK.

Notes: 1. All specifications are in this folder.

2. It is also possible to drag a directory and drop it in the "`include directories [-ada-include-dir]`" part (top left of PolySpace Client) without using the "`Please select a file`" window.

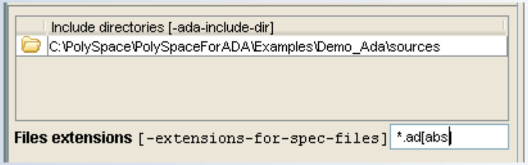
Then, you will still get one last compilation error:



```
*************************************************************
***
*** Ada 95 compliance verifying
***
*************************************************************

Command : rte-fel -feT0 -feS -fe95 -feVf -fea -quiet -fed3 -fedx -feh -fec -fet -feo

Verifying _pst_main
Verifying runtime_error
-> Verifier found an error in example.adb:23:14: "runtime_error (body)" depends on "runtime_error (spec)"
-> Verifier found an error in example.adb:23:14: "runtime_error (spec)" depends on "sensitivity (spec)"
-> Verifier found an error in example.adb:23:14: "sensitivity (spec)" depends on "pkutil (spec)"


----------------------------------------------------------------------
```

```
Verification completed
```

It means that the "`pkutil`" specification is missing from the `-ada-include-dir` directory added just above. By searching for "`pkutil`" specification in the "`sources`" directory, we can see that it is defined in the "`util.adb`" file. By changing the option "Files extension" of "`-extensions-for-specs-file •.ad[sab]`" we can indicate that specifications can be found in `•.ada`, `•.ads` and `•.adb` files:

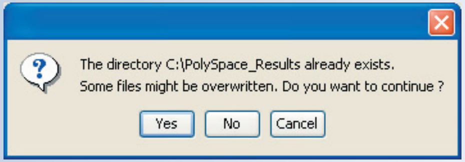

Include directories [-ada-include-dir]
C:\PolySpace\PolySpaceForADA\Examples\Demo_Ada\sources

Files extensions [-extensions-for-spec-files] *.ad[abs]

2.3.2. Progression of the analysis

➤ Click on Execute to restart the analysis.

Some results may have already been written in the "`C:\PolySpace_Results`" directory, because of a previous click on Execute. Therefore a window opens to check whether you want to overwrite them:

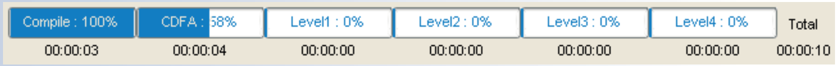

The directory C:\PolySpace_Results already exists.
Some files might be overwritten. Do you want to continue ?

Yes No Cancel

In our example, this is what we want to do. Click on Yes

Note: closing the PolySpace Client window will not stop the PolySpace analysis. If you wish to stop it, click on Stop Execution (you will be asked for confirmation). If the window is closed without stopping the analysis, the analysis continues in the background. Opening again PolySpace Client with the same project automatically updates the analysis with its current status.

The progress bar allows following the progress of the analysis:

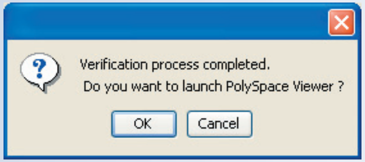


Compile : 100%	CDFA : 58%	Level1 : 0%	Level2 : 0%	Level3 : 0%	Level4 : 0%	Total
00:00:03	00:00:04	00:00:00	00:00:00	00:00:00	00:00:00	00:00:10

A progress report may be obtained by clicking on Compile Log for the compilation phase, or Full Log for the full analysis in the bottom part of the window. Click on Stats to get additional information about the current analysis (list of options, stubbed functions, functions used during main construction, checks found after each phase, etc.). Click on the icon to refresh the summary.

2.3.3. End of the analysis

When the analysis ends, PolySpace proposes to review the results:



Verification process completed.
Do you want to launch PolySpace Viewer ?

➤ Click on OK and go to next section of the tutorial to view the results.

If you click on OK, and if no other analysis is running, you can access the results via the icon in the title bar.

3. Step 2:

PolySpace Viewer - Exploration of results

This step illustrates how to explore analysis results that were generated by either PolySpace Client or PolySpace Server. We review the results of the analysis of "`example.adb`" performed during Step 1 using the following icon:

If the OK button has been clicked at the end of the previous analysis (see previous section), PolySpace Viewer automatically opens results.

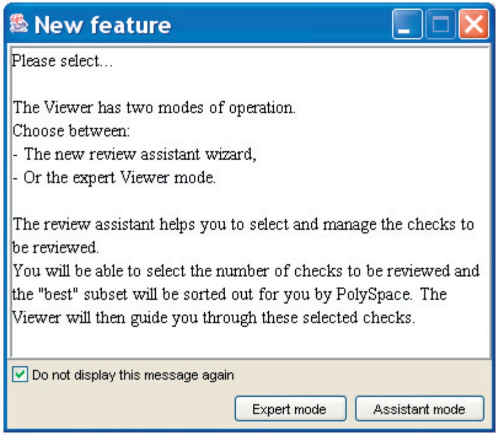
3.1. PolySpace Viewer modes of operation

The first time the PolySpace Viewer is opened, a window will appear to describe the different modes of operation.

- In "`Expert mode`", all checks can be seen. The number and categories of checks to be reviewed as well as the order in which to review them can be chosen by the user (See next section).
- In "`Assistant mode`", the rules of the review follows a methodology selected by PolySpace. The "best" subset of checks will be automatically selected and sorted out. The PolySpace Viewer will then guide the user through these selected checks.

➤ For this tutorial, please untick "`Do not display this message again`" and then click on "`Expert mode`".

Note: The mode of operation may be changed later in PolySpace Viewer.


New feature

Please select...

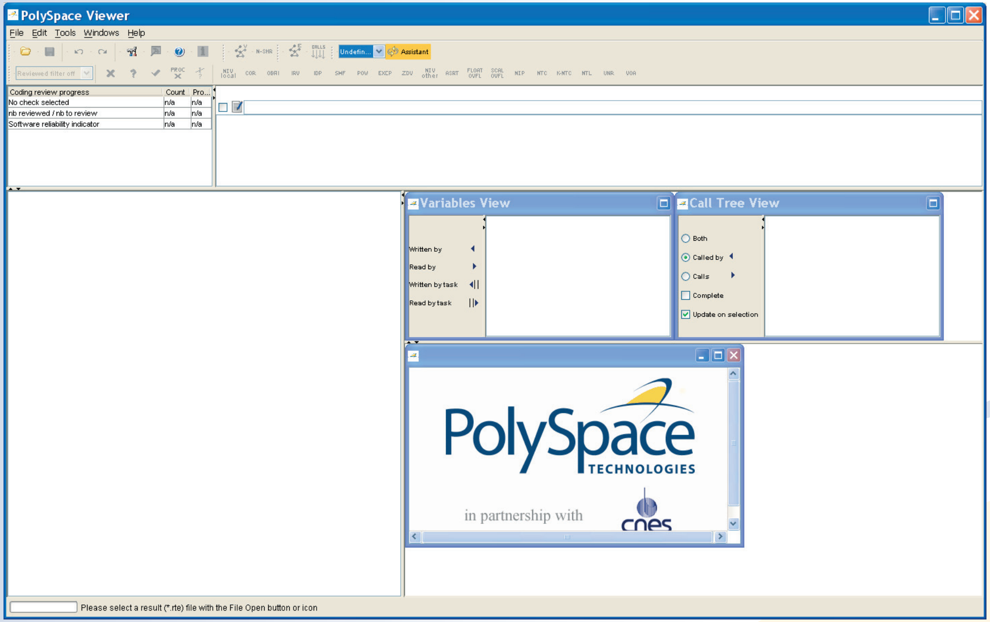
The Viewer has two modes of operation.
Choose between:
- The new review assistant wizard,
- Or the expert Viewer mode.

The review assistant helps you to select and manage the checks to be reviewed.
You will be able to select the number of checks to be reviewed and the "best" subset will be sorted out for you by PolySpace. The Viewer will then guide you through these selected checks.

✓ Do not display this message again

3.2. Download results into the Viewer

After having clicked on "`Expert mode`" the PolySpace Viewer window looks like below:

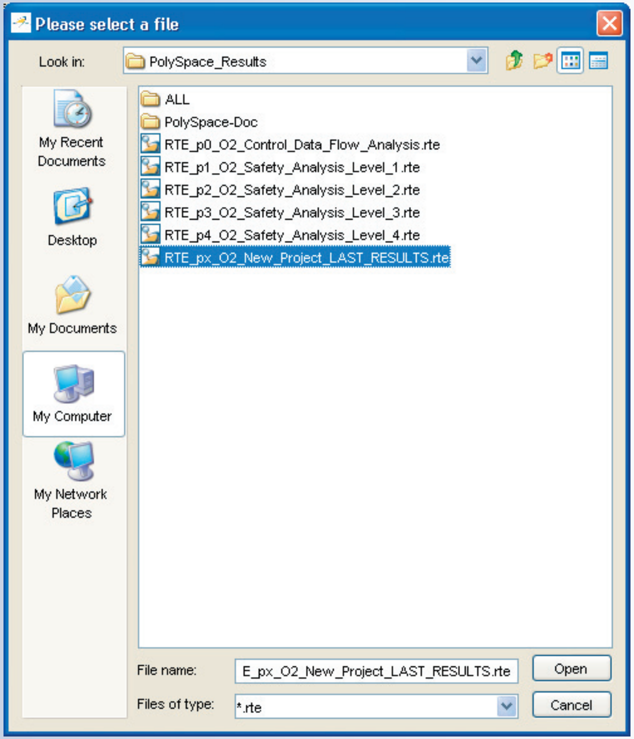


Coding review progress	Count	Pro...
No check selected	n/a	n/a
nb reviewed / nb to review	n/a	n/a
Software reliability indicator	n/a	n/a

○ Both [✓ ● Called by] ○ Calls ☐ Complete [✓ Update on selection]

Please select a result (*.rte) file with the File Open button or icon

➤ Click "`File>Open`" to load your result files. If you did not perform the analysis, you can still review the results by opening the following file:
`<PolySpaceInstallDir>\Examples\Demo_Ada\RTE_px_O2_Demo_Ada_LAST_RESULTS.rte`


Please select a file

Look in: PolySpace_Results

ALL
PolySpace-Doc
RTE_p0_O2_Control_Data_Flow_Analysis.rte
RTE_p1_O2_Safety_Analysis_Level_1.rte
RTE_p2_O2_Safety_Analysis_Level_2.rte
RTE_p3_O2_Safety_Analysis_Level_3.rte
RTE_p4_O2_Safety_Analysis_Level_4.rte
RTE_px_O2_New_Project_LAST_RESULTS.rte

File name: E_px_O2_New_Project_LAST_RESULTS.rte

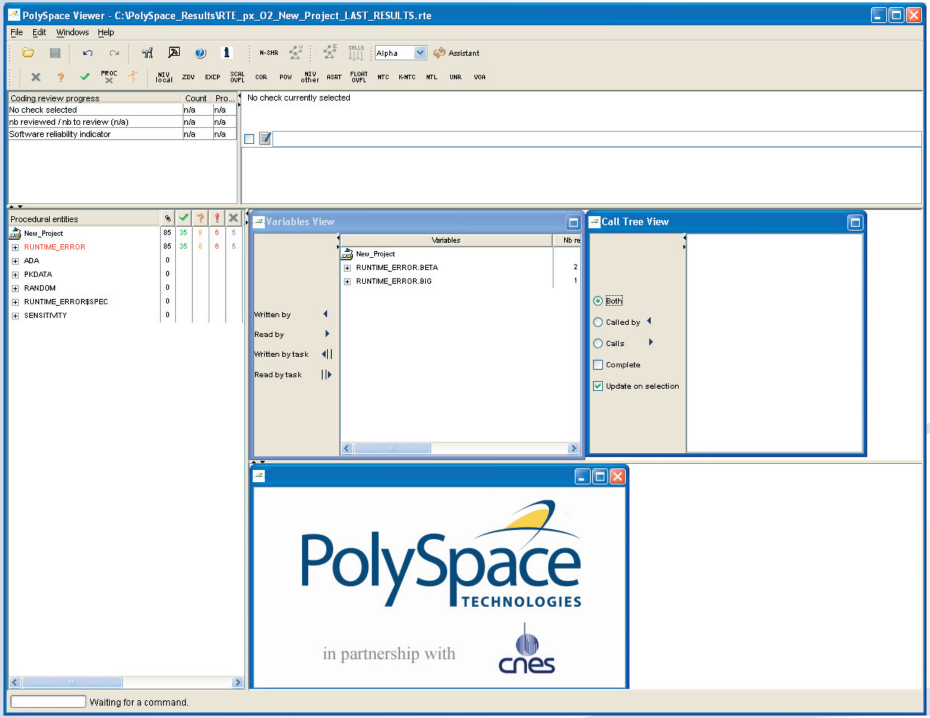
Files of type: *.rte

➤ Then click on Open to proceed with further steps

Note: The `RTE_px_O2_Demo_Ada_LAST_RESULTS.rte` is a "link" to the best results for the analysis in term of precision: `RTE_p4_O2_Safety_Analysis_Level4.rte`. Other results have lower precision.

3.3. Reviewing PolySpace results in "Expert mode" ("`example.adb`")

After loading the results in "`Expert`" mode, PolySpace Viewer window looks like below:



Coding review progress	Count	Pro...
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

No check currently selected

Procedural entities	%	✓	?	!	✕
New_Project	85	35	8	6	5
RUNTIME_ERROR	85	35	8	6	5
ADA	0				
PKDATA	0				
RANDOM	0				
RUNTIME_ERROR$SPEC	0				
SENSITIVITY	0				

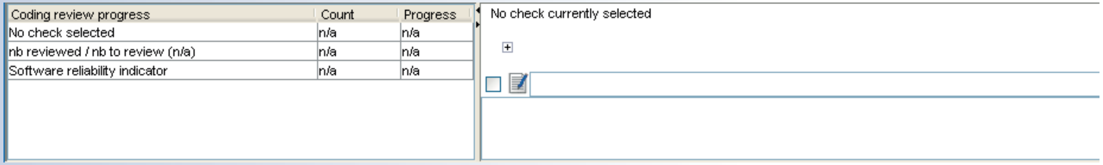
Variables	Nb re
New_Project	
RUNTIME_ERROR.BETA	2
RUNTIME_ERROR.BIG	1

Waiting for a command.

1. On the left is the run-time error view (RTE or "`Procedural Entities`" View).
   It displays the list of packages which have been analysed or used during the analysis (specifications).

2. In the bottom right area is the source code view. Each operation checked is displayed using meaningful colour scheme and related diagnostic:
   - Red: Errors which occur at every execution.
   - Orange: Unproven - an error may occur sometimes.
   - Grey: Shows unreachable code.
   - Green: Error condition that will never occur.

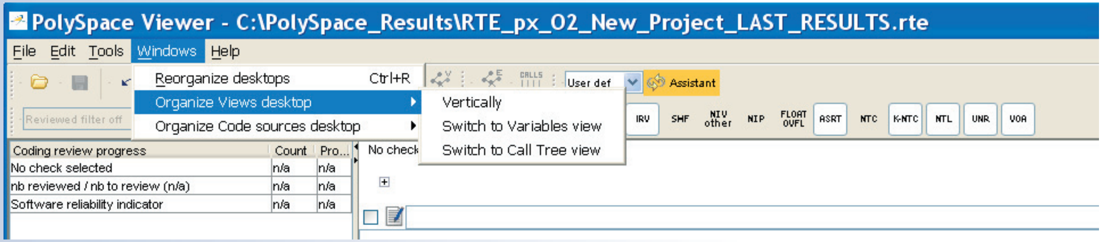
3. The two windows just below the tool bar display details about the currently reviewed check (when the check has been selected):



Coding review progress	Count	Progress
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

No check currently selected

4. The top right area is used for displaying both control and data flow results.
   You can switch from one view to the other by using the "`Windows`" menu:



PolySpace Viewer - C:\PolySpace_Results\RTE_px_02_New_Project_LAST_RESULTS.rte

Coding review progress	Count	Pro...
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

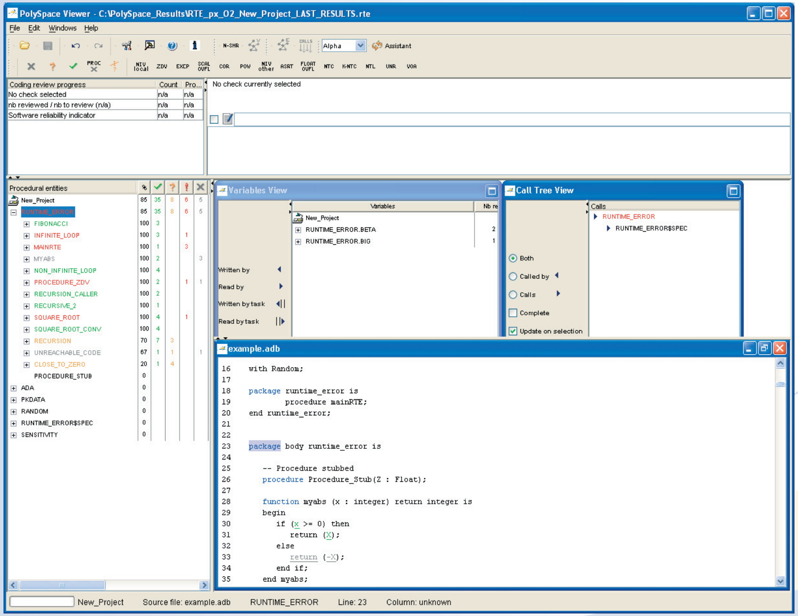
No check

3.3.1. Procedural Entities view (RTE view)

Each package and underlying functions in the RTE view is colorized according to the most critical error found:

- In black color: The packages specification has been used to perform analysis
- In red color: The package is red: one or more *definite* run-time errors have been found in it.

➤ Click once on the ⊞ left of "RUNTIME_ERROR" to find out more about this package.


PolySpace Viewer - C:\PolySpace_Results\RTE_px_02_New_Project_LAST_RESULTS.rte
File Edit Windows Help
Alpha
Assistant
N-SHR
NIV local ZDV EXEP SCAL OVFL COR POW NIV other ASRT FLOAT OVFL NTC K-NTC NTL UNR VOA
Variables View
Call Tree View
Written by
Read by
Written by task
Read by task
Both
Called by
Calls
Complete
Update on selection

Coding review progress	Count	Pro...
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

No check currently selected

Procedural entities	%	✓	?	!	✗
New_Project	85	35	8	6	5
⊟ RUNTIME_ERROR	85	35	8	6	5
⊞ FIBONACCI	100	3			
⊞ INFINITE_LOOP	100	3		1	
⊞ MAINRTE	100	1		3	
⊞ MYABS	100	2			3
⊞ NON_INFINITE_LOOP	100	4			
⊞ PROCEDURE_ZDV	100	2		1	1
⊞ RECURSION_CALLER	100	2			
⊞ RECURSIVE_2	100	1			
⊞ SQUARE_ROOT	100	4		1	
⊞ SQUARE_ROOT_CONV	100	4			
⊞ RECURSION	70	7	3		
⊞ UNREACHABLE_CODE	67	1	1		1
⊞ CLOSE_TO_ZERO	20	1	4		
PROCEDURE_STUB	0				
⊞ ADA	0				
⊞ PKDATA	0				
⊞ RANDOM	0				
⊞ RUNTIME_ERROR$SPEC	0				
⊞ SENSITIVITY	0				

Variables	Nb re
New_Project	
⊞ RUNTIME_ERROR.BETA	2
⊞ RUNTIME_ERROR.BIG	1

Calls
- RUNTIME_ERROR
  - RUNTIME_ERROR$SPEC

example.adb

```
with Random;

package runtime_error is
        procedure mainRTE;
end runtime_error;


package body runtime_error is

   -- Procedure stubbed
   procedure Procedure_Stub(Z : Float);

   function myabs (x : integer) return integer is
   begin
      if (x >= 0) then
         return (X);
      else
         return (-X);
      end if;
   end myabs;
```

New_Project   Source file: example.adb   RUNTIME_ERROR   Line: 23   Column: unknown

"RUNTIME_ERROR" is expanded and the list of functions defined within "RUNTIME_ERROR" is displayed.
The functions in red or grey (PROCEDURE_ZDV, SQUARE_ROOT, etc.) have code sections that need to be inspected first because they are definite diagnosis of PolySpace (either runtime errors or dead code).

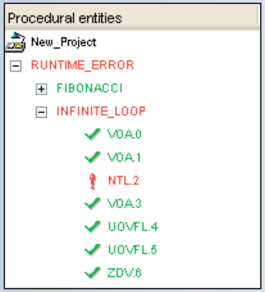
The columns (%, !, ✓, ✗, ?,...) provide information about run-time errors found in each function:

- The % column indicates the selectivity (level of proof),
- The ! column indicates the number of definite run-time errors or reds,
- The ? column indicates the number of unproven or oranges (may be run-time errors that do not occur systematically),
- The ✓ column indicates the number of safe operations or greens
- The ✗ column indicates the number of unreachable instructions or grey code sections.

Let's have a look at some errors found by PolySpace in the analyzed package.

First example of runtime error found by PolySpace: Memory Corruption

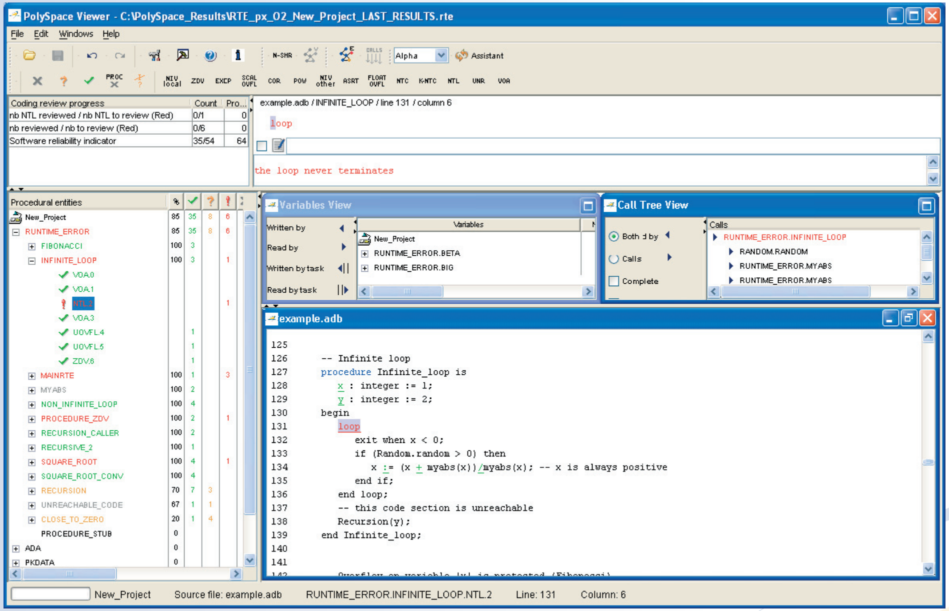
➤ Click on ⊞ to expand "`Pointer_Arithmetic()`" to find out more about the red error.
It displays a list of red, green, and orange symbols, featuring the complete list of code areas that PolySpace checked within the "`Pointer_Arithmetic()`" function.



```
Procedural entities
New_Project
⊟ RUNTIME_ERROR
    ⊞ FIBONACCI
    ⊟ INFINITE_LOOP
        ✓ VOA.0
        ✓ VOA.1
        ! NTL.2
        ✓ VOA.3
        ✓ UOVFL.4
        ✓ UOVFL.5
        ✓ ZDV.6
```

➤ Click on the red "`NTL.0`" item - which stands for **N**on-**T**ermination of **L**oop -, to precisely locate this error in the source code. The bottom right section is updated showing the location of the "`NTL.0`" item.

Step 2: PolySpace Viewer - Exploration of results



Coding review progress	Count	Pro...
nb NTL reviewed / nb NTL to review (Red)	0/1	0
nb reviewed / nb to review (Red)	0/6	0
Software reliability indicator	35/54	64

example.adb / INFINITE_LOOP / line 131 / column 6

```
loop
```

```
the loop never terminates
```

Procedural entities	%	✓	?	!
New_Project	85	35	8	6
RUNTIME_ERROR	85	35	8	6
FIBONACCI	100	3		
INFINITE_LOOP	100	3		1
✓ VOA.0				
✓ VOA.1				
! NTL.2				1
✓ VOA.3				
✓ UOVFL.4		1		
✓ UOVFL.5		1		
✓ ZDV.6		1		
MAINRTE	100	1		3
MYABS	100	2		
NON_INFINITE_LOOP	100	4		
PROCEDURE_ZDV	100	2		1
RECURSION_CALLER	100	2		
RECURSIVE_2	100	1		
SQUARE_ROOT	100	4		1
SQUARE_ROOT_CONV	100	4		
RECURSION	70	7	3	
UNREACHABLE_CODE	67	1	1	
CLOSE_TO_ZERO	20	1	4	
PROCEDURE_STUB	0			
ADA	0			
PKDATA	0			

Written by
Read by
Written by task
Read by task

Variables
New_Project
RUNTIME_ERROR.BETA
RUNTIME_ERROR.BIG

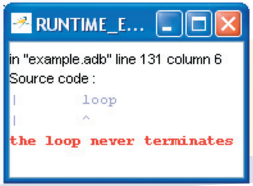
Both by
Calls
Complete

Calls
RUNTIME_ERROR.INFINITE_LOOP
RANDOM.RANDOM
RUNTIME_ERROR.MYABS
RUNTIME_ERROR.MYABS

```

   -- Infinite loop
   procedure Infinite_loop is
      x : integer := 1;
      y : integer := 2;
   begin
      loop
         exit when x < 0;
         if (Random.random > 0) then
            x := (x + myabs(x))/myabs(x); -- x is always positive
         end if;
      end loop;
      -- this code section is unreachable
      Recursion(y);
   end Infinite_loop;


```

New_Project   Source file: example.adb   RUNTIME_ERROR.INFINITE_LOOP.NTL.2   Line: 131   Column: 6

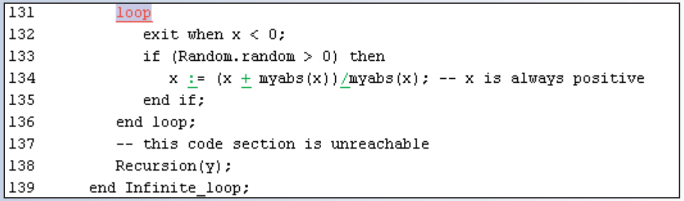
➤ Click on the red loop in the source code at line 121.
An error message is opened:



in "example.adb" line 131 column 6
Source code :

```
|        loop
|        ^
the loop never terminates
```

Indeed, the condition to exit the loop is that "x" becomes a negative value.

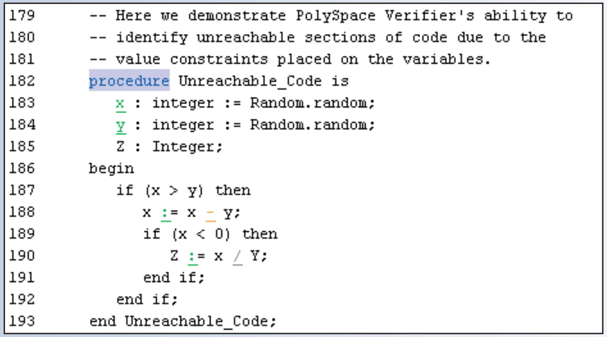


```
   loop
      exit when x < 0;
      if (Random.random > 0) then
         x := (x + myabs(x))/myabs(x); -- x is always positive
      end if;
   end loop;
   -- this code section is unreachable
   Recursion(y);
end Infinite_loop;
```

But in the current situation, "x" will always be a positive value. So, the loop can't terminate.

Second example of runtime error found by PolySpace: Unreachable code

➤ Select "UNREACHABLE_CODE" in the RTE View. You can see that the division "z := x / y" is unreachable (gray colour on the check) because of the non satisfied boolean condition: "x" is never negative when evaluating "x<0". PolySpace has detected some dead code.



```
-- Here we demonstrate PolySpace Verifier's ability to
-- identify unreachable sections of code due to the
-- value constraints placed on the variables.
procedure Unreachable_Code is
   x : integer := Random.random;
   y : integer := Random.random;
   Z : Integer;
begin
   if (x > y) then
      x := x - y;
      if (x < 0) then
         Z := x / Y;
      end if;
   end if;
end Unreachable_Code;
```

3.3.2. Colours in the source code view

Each operation checked is also displayed using meaningful colour scheme and related diagnostic in the source code view as links:

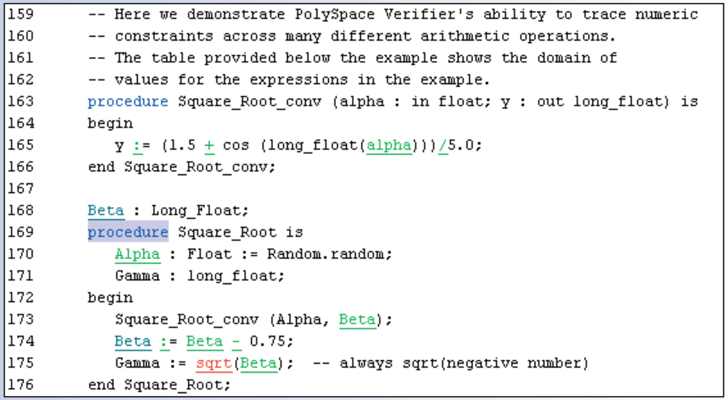
- Red: A link to the error message associated to the error which occurs at every execution.
- Orange: A link to a warning message - an error may occur sometimes.
- Grey: A link to a check shown as unreachable code. The error message is in grey.
- Green: A link to a VOA (Value on Assignment) or an error condition that will never occur.
- Black: represents some comments, source code that does not contain any operation to be checked by PolySpace in terms of run time errors and optimized operations, e.g. `x := 0;`.
- Blue: text highlighting the keyword "`procedure`" and "`function`".
- Underlined blue: A link to a global variable in the "Global variable View".

3.3.3. More examples of run-time errors

Unlike most other testing techniques, PolySpace provides the benefit of finding the exact location of run-time errors in the source code. Below are some examples that you can review with PolySpace Viewer.

In an First example of the second set: Arithmetic error

➤ Click on ⊞ to expand "`SQUARE_ROOT`" function. You can see the source code view in the bottom right. You can also display the call tree for that function by using the "Windows" menu (see previous paragraph).
"`SQUARE_ROOT`" is called by `MAINRTE` function. It is displayed in the "*Call tree view*" window
(right of the top right section).
"`SQUARE_ROOT`" calls "`RANDOM.random`" (automatically stubbed function), "`SQUARE_ROOT_CONV`" (from RUNTIME_ERROR package) and "`SQRT`" (from the standard library).

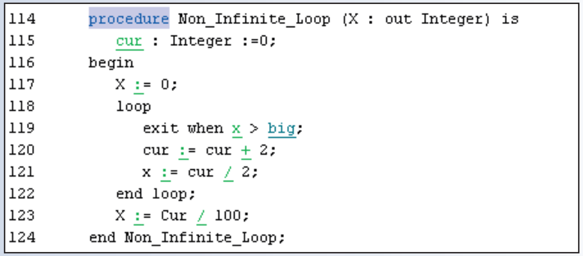

```
-- Here we demonstrate PolySpace Verifier's ability to trace numeric
-- constraints across many different arithmetic operations.
-- The table provided below the example shows the domain of
-- values for the expressions in the example.
procedure Square_Root_conv (alpha : in float; y : out long_float) is
begin
   y := (1.5 + cos (long_float(alpha)))/5.0;
end Square_Root_conv;

Beta : Long_Float;
procedure Square_Root is
   Alpha : Float := Random.random;
   Gamma : long_float;
begin
   Square_Root_conv (Alpha, Beta);
   Beta := Beta - 0.75;
   Gamma := sqrt(Beta);  -- always sqrt(negative number)
end Square_Root;
```

The green sections into the source code view are error-free but the red (sqrt) is an issue that needs to be fixed. Indeed, when the local float variable gamma is computed in the line "gamma=sqrt(beta - 0.75);", the operation will cause a run-time error, as the parameter passed to "sqrt" is always negative.

Note: using the -voa option when launching the analysis, PolySpace can give more information about the range on scalar assignments

Second example of the second set: Non-Infinite loop

➤ Select "`NON_INFINITE_LOOP`" in RTE View. The function is fully green: it means that the locale variable x never overflows, even if the `exit` condition of loop deals with `y` that is smaller than `x`. PolySpace confirms that the function always terminates.



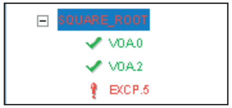
```
procedure Non_Infinite_Loop (X : out Integer) is
   cur : Integer :=0;
begin
   X := 0;
   loop
      exit when x > big;
      cur := cur + 2;
      x := cur / 2;
   end loop;
   X := Cur / 100;
end Non_Infinite_Loop;
```

3.3.4. Advanced results exploration

You can filter the information provided by PolySpace to focus on the type of errors you wish to investigate. There are pre-defined composite filters "Alpha", "Beta", "User Def" and "Gamma" that you can choose depending on your development process.

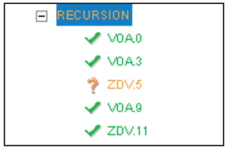
➤ Click on the Gamma button to get all the "red" and "grey" code sections. It is mainly used during the earliest development stages to focus quickly on critical bugs.

➤ To illustrate the use of these filters, we will focus on the Square Root function that we have examined in the 3.3.2 section. Having clicked on "Gamma" reduced the number of checks related to "SQUARE_ROOT".


SQUARE_ROOT
✓ VOA.0
✓ VOA.2
! EXCP.5

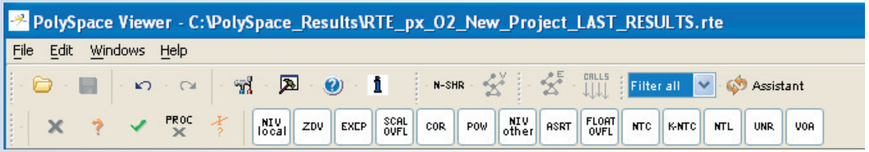
The list of acronyms - for type of operations checked - shows what kind of errors PolySpace automatically looked for.

➤ Select the "Beta" mode (which is the default mode). It highlights checks that could cause a processor halt, memory corruptions or overflows. Select again "RECURSION" in the "Procedural entities" view and then, click on ⊞ to expand it and get the list of checks.


RECURSION
✓ VOA.0
✓ VOA.3
? ZDV.5
✓ VOA.9
✓ ZDV.11

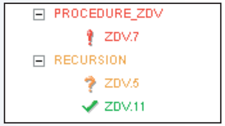
To get the comprehensive list of operations checked by PolySpace, you can switch to "Alpha" mode. "Undefined" is selected when switching between expert and assistant mode.

You may also want to use filters to focus on particular categories of errors. Those filters are located at the top of the PolySpace Viewer window:

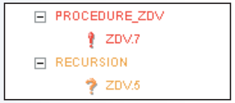

PolySpace Viewer - C:\PolySpace_Results\RTE_px_02_New_Project_LAST_RESULTS.rte

Notes: When the mouse pointer moves on a filter, a tool tip gives its definition.

➤ Click on "`Filter All`" (top of the window) to suppress all checks and click on ZDV.
You will get a list of checks containing only ZDV (Zero DiVision) reds, oranges or greens:


```
⊟ PROCEDURE_ZDV
      ZDV.7
⊟ RECURSION
      ZDV.5
      ZDV.11
```

➤ Click on "`VOA`" and ✓ (top of the window) to suppress green code sections.
You will get a reduced list of checks reds, oranges and grays:


```
⊟ PROCEDURE_ZDV
      ZDV.7
⊟ RECURSION
      ZDV.5
```

3.3.5. Miscellaneous

The icon gives access to the PolySpace Manual. All views have a pop-up menu (right click on mouse).

➤ Close the PolySpace Viewer window by clicking on the upper right ☒ symbol (PolySpace Viewer can also be closed using "`File>Close`").

3.3.6. Methodological Assistant

After this first usage of PolySpace Viewer, some simple questions remain:

- Do all checks need to be reviewed?
- What are the checks to review?
- How many?
- What is the best order?

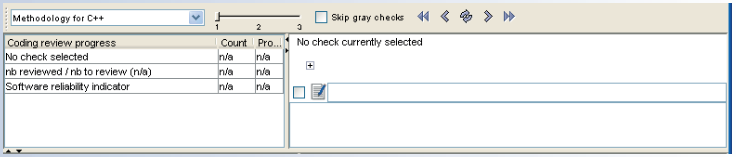
The Methodological assistant answers to all theses questions. It helps to select and manage the checks to be reviewed. It selects a "best" subset and sorts checks out. The Assistant mode in the PolySpace Viewer will then guide you through these selected checks.

➤ If the PolySpace Viewer is still open, close it by clicking on the upper right symbol, open it again, load the same results and chose the "`Assistant`" mode.

After having loaded the results in "`Assistant`" mode, PolySpace Viewer window looks like below.

Assistant dashboard

The second line of buttons on the toolbar and the two views just below are used to navigate between the checks selected by PolySpace:



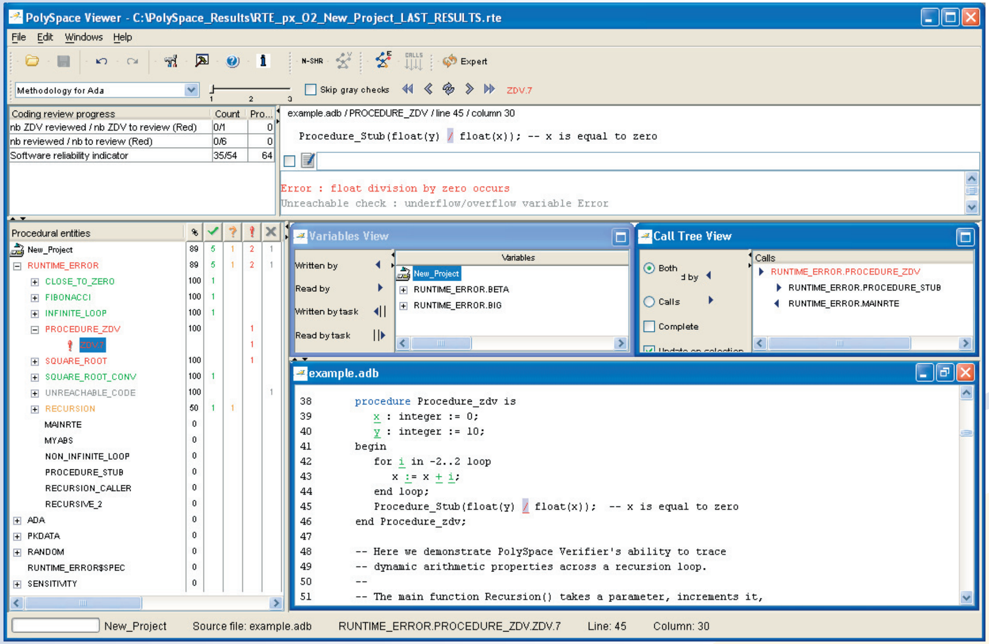
Coding review progress	Count	Pro...
No check selected	n/a	n/a
nb reviewed / nb to review (n/a)	n/a	n/a
Software reliability indicator	n/a	n/a

No check currently selected

PolySpace Viewer has also been updated as follows:

1. Now, in the "Procedural Entities" view the list of files analyzed is sorted out according to the methodological assistant used.
2. In the source code view, each operation will be sorted out according to the PolySpace methodology in the following order:
   - Red: The methodological assistant browses all red errors.
   - Gray: The methodological assistant browses unreachable code depending on the radio button "`Skip gray checks`".
   - Orange: The methodological assistant chooses and reviews the "best" unproven operations - those that are the most probably actual errors.

➤ Click on to navigate to the next unreviewed check.
PolySpace Viewer has been refreshed with the first check selected by the methodological assistant:



PolySpace Viewer - C:\PolySpace_Results\RTE_px_02_New_Project_LAST_RESULTS.rte

Methodology for Ada — Skip gray checks — ZDV.7

Coding review progress	Count	Pro...
nb ZDV reviewed / nb ZDV to review (Red)	0/1	0
nb reviewed / nb to review (Red)	0/6	0
Software reliability indicator	35/54	64

example.adb / PROCEDURE_ZDV / line 45 / column 30

```
Procedure_Stub(float(y) / float(x)); -- x is equal to zero
```

```
Error : float division by zero occurs
Unreachable check : underflow/overflow variable Error
```

Procedural entities	%	✓	?	!	✗
New_Project	89	5	1	2	1
RUNTIME_ERROR	89	5	1	2	1
CLOSE_TO_ZERO	100	1			
FIBONACCI	100	1			
INFINITE_LOOP	100	1			
PROCEDURE_ZDV	100			1	
ZDV.7				1	
SQUARE_ROOT	100			1	
SQUARE_ROOT_CONV	100	1			
UNREACHABLE_CODE	100				1
RECURSION	50	1	1		
MAINRTE	0				
MYABS	0				
NON_INFINITE_LOOP	0				
PROCEDURE_STUB	0				
RECURSION_CALLER	0				
RECURSIVE_2	0				
ADA	0				
PKDATA	0				
RANDOM	0				
RUNTIME_ERROR$SPEC	0				
SENSITIVITY	0				

example.adb

```
38      procedure Procedure_zdv is
39         x : integer := 0;
40         y : integer := 10;
41      begin
42         for i in -2..2 loop
43            x := x + i;
44         end loop;
45         Procedure_Stub(float(y) / float(x));  -- x is equal to zero
46      end Procedure_zdv;
47
48      -- Here we demonstrate PolySpace Verifier's ability to trace
49      -- dynamic arithmetic properties across a recursion loop.
50      --
51      -- The main function Recursion() takes a parameter, increments it,
```

New_Project   Source file: example.adb   RUNTIME_ERROR.PROCEDURE_ZDV.ZDV.7   Line: 45   Column: 30

The methodological dashboard gives details and allows reviewing the check. On the selected check, it is possible to mark the fact that it has been reviewed and write an associated comment.

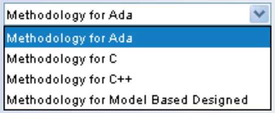
The left part of the dashboard has been updated, and displays some statistics in three lines:

- The first line gives the number and percentage of remaining checks to review in the selected category (here, red ZDV checks).
- The second line gives values for the whole colour category (red, grey and unproven).
- The last line gives values for the whole software being reviewed. This is called `Software reliability indicator`. It gives the percentage of green checks compared to the total number of checks.

Other buttons in the Methodological dashboard allow navigating to the next or previous check that hasn't been reviewed yet. It's also possible to refresh the different views to come back to the check currently being reviewed using the button.

Choose a methodological assistant

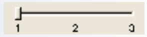
Some methodologies



Methodology for Ada

Methodology for Ada
Methodology for C
Methodology for C++
Methodology for Model Based Designed

and

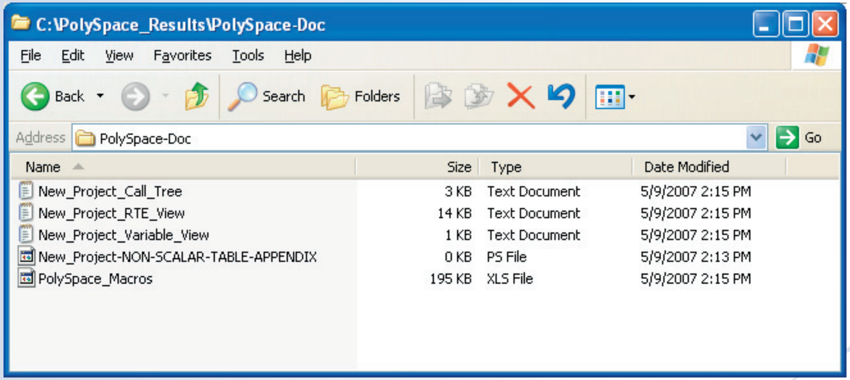


associated levels have been pre-defined by PolySpace.

The methodology allows selecting the categories of checks to review, the number for each category and their order based on statistics on many analysis results.
The level defines the number of checks to review by category. It is chosen according to the development phase during which the code has been analyzed: "`Fresh code`", "`Unit test`" and "`Code review`"
It is possible to define your own Methodology
(From the "`Edit`" menu, Click on "`Preferences... >Assistant methodology`".)
Here, you can create a new configuration set and define for each level what will be the categories of check to review and how many of each category.

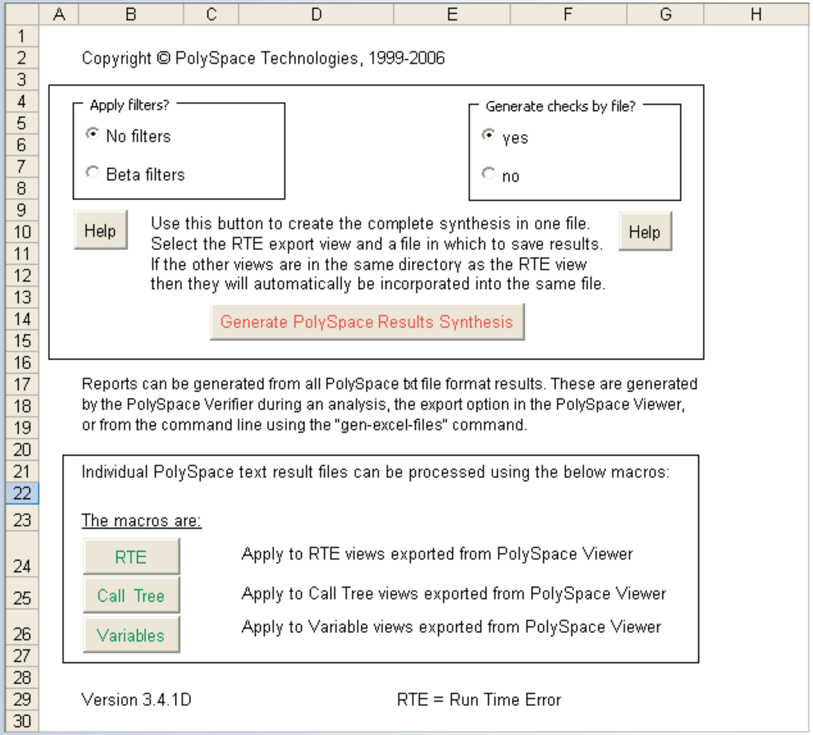
3.4. Report Generation

When PolySpace performs an analysis, it generates textual files that can be used to create Excel® reports.
These files are located in the results directory (See "`C:\PolySpace_Results\PolySpace-Doc`" or "`<PolySpaceInstallDir>\Examples\Demo_Ada\PolySpace-Doc`").
These files contain data related to all views except the source code one.
The "`C:\PolySpace_Results\PolySpace-Doc`" directory should contain the following files:


C:\PolySpace_Results\PolySpace-Doc
File Edit View Favorites Tools Help
Back
Search
Folders
Address PolySpace-Doc
Go

Name	Size	Type	Date Modified
New_Project_Call_Tree	3 KB	Text Document	5/9/2007 2:15 PM
New_Project_RTE_View	14 KB	Text Document	5/9/2007 2:15 PM
New_Project_Variable_View	1 KB	Text Document	5/9/2007 2:15 PM
New_Project-NON-SCALAR-TABLE-APPENDIX	0 KB	PS File	5/9/2007 2:13 PM
PolySpace_Macros	195 KB	XLS File	5/9/2007 2:15 PM

➤ Open the file called "`PolySpace_Macros.xls`" and enable macros to display the Excel® file below::



Copyright © PolySpace Technologies, 1999-2006

Apply filters?
- [✓ No filters]
- Beta filters

Generate checks by file?
- [✓ yes]
- no

Use this button to create the complete synthesis in one file.
Select the RTE export view and a file in which to save results.
If the other views are in the same directory as the RTE view
then they will automatically be incorporated into the same file.

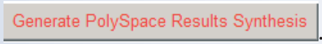
Reports can be generated from all PolySpace txt file format results. These are generated by the PolySpace Verifier during an analysis, the export option in the PolySpace Viewer, or from the command line using the "gen-excel-files" command.

Individual PolySpace text result files can be processed using the below macros:

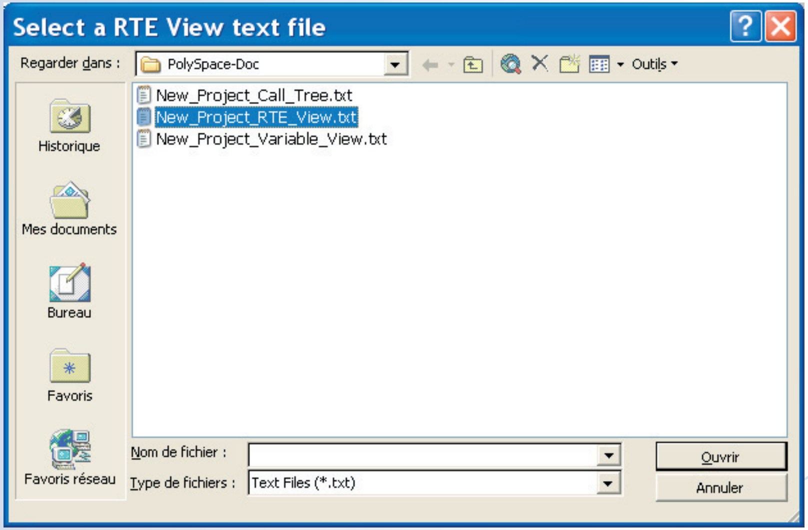
The macros are:

RTE	Apply to RTE views exported from PolySpace Viewer
Call Tree	Apply to Call Tree views exported from PolySpace Viewer
Variables	Apply to Variable views exported from PolySpace Viewer

Version 3.4.1D   RTE = Run Time Error

➤ Click on 

. A file browser opens. Select the file called "New_Project_RTE_View.txt" as shown below:



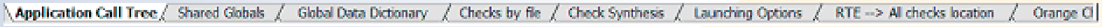
Select a RTE View text file

Regarder dans : PolySpace-Doc

New_Project_Call_Tree.txt
New_Project_RTE_View.txt
New_Project_Variable_View.txt

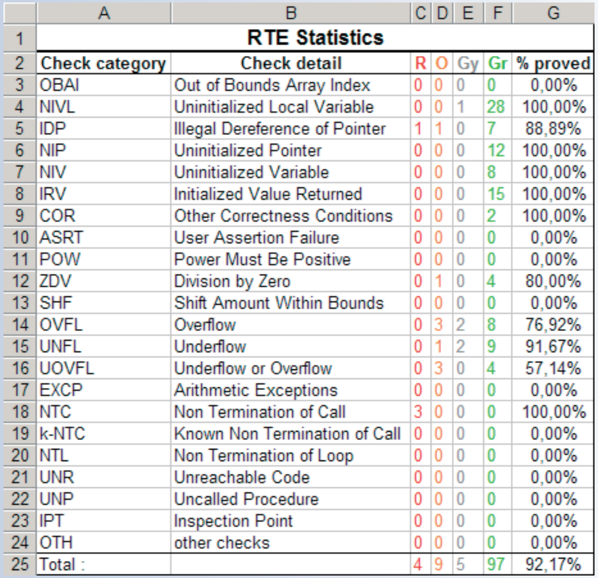
Nom de fichier :
Type de fichiers : Text Files (*.txt)

After a few seconds, an Excel® file is generated. It contains several spreadsheets related to the application analyzed.



Application Call Tree / Shared Globals / Global Data Dictionary / Checks by file / Check Synthesis / Launching Options / RTE --> All checks location / Orange Cl

For example, in "`Checks Synthesis`" all statistics about checks and colors are reported in a summary table.



	A	B	C	D	E	F	G
1	**RTE Statistics**						
2	**Check category**	**Check detail**	**R**	**O**	**Gy**	**Gr**	**% proved**
3	OBAI	Out of Bounds Array Index	0	0	0	0	0,00%
4	NIVL	Uninitialized Local Variable	0	0	1	28	100,00%
5	IDP	Illegal Dereference of Pointer	1	1	0	7	88,89%
6	NIP	Uninitialized Pointer	0	0	0	12	100,00%
7	NIV	Uninitialized Variable	0	0	0	8	100,00%
8	IRV	Initialized Value Returned	0	0	0	15	100,00%
9	COR	Other Correctness Conditions	0	0	0	2	100,00%
10	ASRT	User Assertion Failure	0	0	0	0	0,00%
11	POW	Power Must Be Positive	0	0	0	0	0,00%
12	ZDV	Division by Zero	0	1	0	4	80,00%
13	SHF	Shift Amount Within Bounds	0	0	0	0	0,00%
14	OVFL	Overflow	0	3	2	8	76,92%
15	UNFL	Underflow	0	1	2	9	91,67%
16	UOVFL	Underflow or Overflow	0	3	0	4	57,14%
17	EXCP	Arithmetic Exceptions	0	0	0	0	0,00%
18	NTC	Non Termination of Call	3	0	0	0	100,00%
19	k-NTC	Known Non Termination of Call	0	0	0	0	0,00%
20	NTL	Non Termination of Loop	0	0	0	0	0,00%
21	UNR	Unreachable Code	0	0	0	0	0,00%
22	UNP	Uncalled Procedure	0	0	0	0	0,00%
23	IPT	Inspection Point	0	0	0	0	0,00%
24	OTH	other checks	0	0	0	0	0,00%
25	Total :		4	9	5	97	92,17%

4. Launch PolySpace Remotely

This paragraph describes the basic steps to launch an analysis in remote. To do so you need:

1. A Queue Manager server (QM) installed.
2. Your desktop PC configured with a PolySpace Client.
3. A networked machine configured with a PolySpace Server.

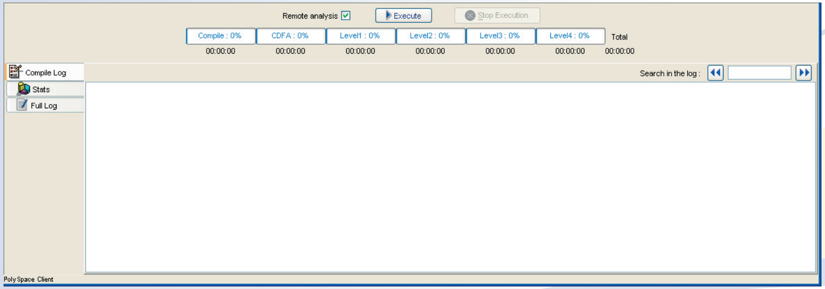
Please see the PolySpace Installation guide (available on the PolySpace CD-ROM in `\Docs\Install`) to install and configure, the Queue Manager, a Client and a Server.

Note: Launching an analysis remotely requires a PolySpace Server product and associated license.

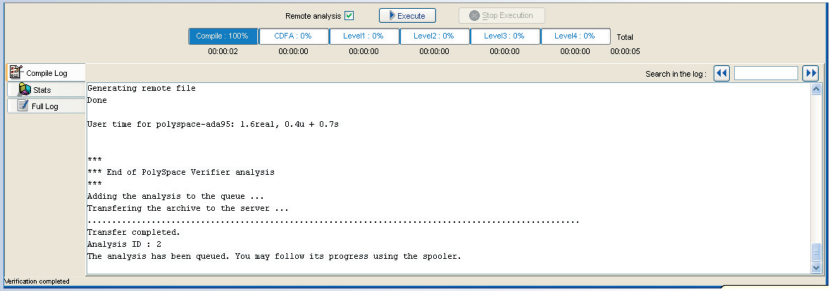
4.1. Launching an analysis

It can be done in two steps:

➤ Step 1: set up an analysis as described at step 1 but do not launch it.
➤ Step 2: tick the "`Remote analysis`" checkbox (see figure below) and click on Execute to launch the analysis.



The analysis starts and the compilation phase is performed on the desktop PC.
At the end of the "Compilation phase" the analysis is sent to the Queue Manager server.
By clicking on the "Full Log" tab, you will see a message like this:



```
Generating remote file
Done

User time for polyspace-ada95: 1.6real, 0.4u + 0.7s


***
*** End of PolySpace Verifier analysis
***
Adding the analysis to the queue ...
Transfering the archive to the server ...
.....................................................................................................
Transfer completed.
Analysis ID : 2
The analysis has been queued. You may follow its progress using the spooler.
```

The analysis has been queued with an ID number, and you can follow its progression using the PolySpace Spooler.

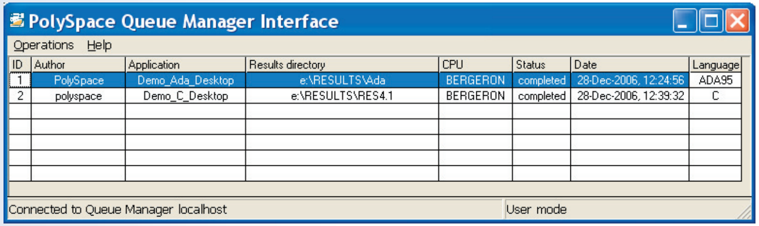
If you do not tick the "Remote analysis" checkbox, the analysis continues locally.

4.2. Management of PolySpace analysis in remote: the PolySpace Spooler

You can check the analysis processes in the queue by clicking on the short cut on your desktop PC

PolySpace Spooler
Raccourci
2 Ko

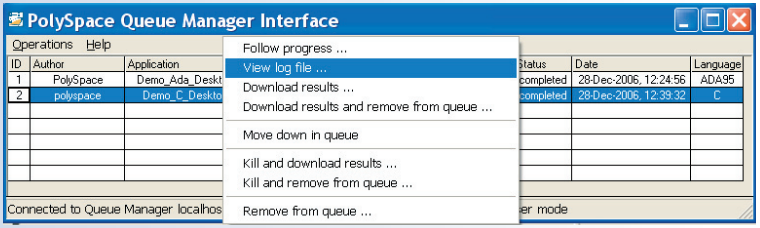
or on the icon in the menu tab of the launcher.



ID	Author	Application	Results directory	CPU	Status	Date	Language
1	PolySpace	Demo_Ada_Desktop	e:\RESULTS\Ada	BERGERON	completed	28-Dec-2006, 12:24:56	ADA95
2	polyspace	Demo_C_Desktop	e:\RESULTS\RES4.1	BERGERON	completed	28-Dec-2006, 12:39:32	C

Connected to Queue Manager localhost
User mode

When you select an analysis and right click, you can manage it in the queue:


PolySpace Queue Manager Interface

Operations Help

ID	Author	Application	Status	Date	Language
1	PolySpace	Demo_Ada_Deskt	completed	28-Dec-2006, 12:24:56	ADA95
2	polyspace	Demo_C_Deskto	completed	28-Dec-2006, 12:39:32	C

Connected to Queue Manager localhos ... er mode

Follow progress ...
View log file ...
Download results ...
Download results and remove from queue ...
Move down in queue
Kill and download results ...
Kill and remove from queue ...
Remove from queue ...

- "*Follow progress*" displays the associated log file in a Launcher window. If the analysis is running, you can follow the update of the log file and associated progress bar in real time.
- "*View log file*" displays the associated log file in a "Command prompt" window, in which you can see the last 100 updated lines of the log file in real time. This option is only available when the analysis is running.
- "*Download results*" downloads the results of an analysis to the Client. If the analysis is still running, already available partial results are downloaded on the Client, without disturbing the analysis. This option is not available for a "queued" analysis (that has not yet began).
- "*Move down in queue*" reduces the priority of a "queued" analysis.
- "*Kill and download results*" stops the analysis definitively and the latest available partial results are downloaded. The status of the analysis changes from "running" to "aborted". The analysis remains in the queue.
- "*Kill and remove from queue*" stops the analysis definitively and removes it from the queue. **The results will be lost.**
- "*Remove from queue*" removes a "queued", "aborted" or "completed" analysis. **The results will be lost.**

The queue can also be managed through the "Operations>" menu:

- "Operations>Purge queue" purges the entire queue or purges only completed and aborted analysis. The queue manager administrator password is required.

- "Operations>Change root password" changes administrator password of the queue manager. By default, the password is "administrator".

4.3. Batch commands

• Launch analysis in batch:

A set of commands allow the launching of analyses in batch (under a Cygwin shell on a Windows machine).
All commands begin with the prefix `<PolySpaceCommonDir>/RemoteLauncher/bin/polyspace-remote-`.
Commands available are `polyspace-remote-ada95` and `polyspace-remote-desktop-ada95`.

They are equivalent to the commands with a prefix `<PolySpaceInstallDir>/bin/polyspace-`.
For example, `polyspace-remote-desktop-ada95 -server [<hostname>:[<port>] | auto]` allows the sending of a Ada client analysis remotely.

• Manage analysis in batch:

In batch and on a UNIX platform, a set of commands allow the management of analyses in the queue.
All theses commands begin with the prefix `<PolySpaceCommonDir>/RemoteLauncher/bin/psqueue-`:

- `psqueue-download <ID> <results dir>`: downloads an identified analysis into a results directory. `[-f]` forces download (without interactivity) and `-admin -p <password>` allows administrator to download results. Use `[-server <name>[:port]]` to select a specific Queue Manager. Use `[-v|version]` to indicate release number.
- `psqueue-kill <ID>`: kills an identified analysis.
- `psqueue-purge all|ended`: removes all or finished analyses in the queue.
- `psqueue-dump`: gives the list of all analyses in the queue associated to the default Queue Manager.
- `psqueue-move-down <ID>`: moves down an identified analysis in the queue.
- `psqueue-remove <id>`: removes an identified analysis in the queue.
- `psqueue-get-qm-server`: gives the name of the default Queue Manager.
- `psqueue-progress <ID>`: gives progression of the currently identified and running analysis.

`[-open-launcher]` displays the log in PolySpace launcher graphical user interface. `[-full]` gives full log file.

- `psqueue-set-password <old password> <new password>`: changes administrator password.
- `psqueue-check-config`: checks the configuration of Queue Manager. `[-check-licenses]` checks for licenses only.
- `psqueue-upgrade`: allows upgrading a Client (see `PolySpace Install Guide` in the `<PolySpaceCommonDir>/Docs directory`). `[-list-versions]` gives the list of available releases for upgrade. `[-install-version <version number> [-install-dir <directory>]] [-silent]` allow to install an upgrade in a given directory potentially in silent mode.

Note: `<PolySpaceCommonDir>/RemoteLauncher/bin/psqueue-<command> -h` gives information about all available options for each command.

4.4. Share analyses between accounts

- `analysis-key.txt` **file**

For security reasons, all analyses spooled are owned by the user who sent them. Each analysis has a unique crypted key.

The public part of the key is stored in a file named `analysis-keys.txt` and associated to a user account. On a UNIX account, this file is located in: "`/home/<username>/.PolySpace`".On a Windows account, it is located in: "`C:\Documents and Settings\<username>\Application Data\PolySpace`".

The format of the ASCII file is the following (`^t means` tabulation):

```
<ID of launching> ^t <server name of IP address> ^t <public key>
```

Example

```
1      m120   27CB36A9D656F0C3F84F959304ACF81BF229827C58BE1A15C8123786
2      m120   2860F820320CDD8317C51E4455E3D1A48DCE576F5C66BEEF391A9962
8      m120   2D51FF34D7B319121D221272585C7E79501FBCC8973CF287F6C12FCA
```

When attempting to manage (download, kill and remove, etc.) a particular analysis, the Queue Manager will examine this file and check the associated public key before allowing the action.
If the key does not exist, an error message appears: "`key for analysis <ID> not found`".

So, if user A wants to manage (for example download results of) an analysis sent by user B, user A should edit his own `analysis-key.txt` file and add into it the line corresponding to that analysis in the `analysis-key.txt` file of user B.

• Sharing analyses between projects with a magic key

A magic key allows sharing analyses without taking into account the `<ID>`. It allows having the same key for all analyses launched. The format is the following:

```
0       <Server id>    <your hexadecimal value>
```

All analyses spooled will use this key instead of random one. This would allow any user that has this key in his `analysis-key.txt` file to manage all analyses sent with the magic key..

Note: The magic key only works for analyses launched after it has been set up. Analyses sent before, will keep their auto-generated random keys.

5. Summary

After having followed each step of this tutorial, you are now able to launch an analysis using PolySpace Client and review some results with PolySpace Viewer. All theses activities can be performed locally on your desktop PC or in a Client/Server architecture.

You will find more information on advanced options available with our tools
in "`PolySpace Ada documentation.pdf`" available on the CD-ROM (in \Docs\Manual\)
or by clicking on ❓ in PolySpace tools.

PolySpace TECHNOLOGIES