

# PolySpace Client for C/C++ 7

## Prove the absence of run-time errors in source code

PolySpace Client™ for C/C++, in conjunction with PolySpace Server™ for C/C++, provides code-based verification that proves the absence of overflow, divide by zero, out-of-bounds array access, and other run-time errors in source code without requiring program execution, code instrumentation, or test cases. PolySpace Client for C/C++ uses abstract interpretation techniques to verify code. You can use it to verify handwritten code, generated code, or a combination of the two, before compilation and test.

### Working with PolySpace Client for C/C++

PolySpace Client for C/C++ provides management and visualization capabilities for verifying software components on a desktop computer. It processes file-by-file or class-by-class verification as soon as the source code is written, updated, or generated. When used with PolySpace Server for C/C++, PolySpace Client for C/C++ lets you submit verification jobs to computer clusters.

Using the PolySpace Client for C/C++ command line, graphical user interfaces, or Eclipse™, you can:

- Import source code
- Customize a project by target, cross-compiler, or other options
- Check code for compliance with MISRA C® or JSF++ (Joint Strike Fighter Air Vehicle C++) standards
- Monitor the status of jobs submitted to PolySpace Server for C/C++
- Download data from the server to report or visualize verification results
- Document and log verification results

### KEY FEATURES

- File- and class-level verification of software components
- Abstract interpretation techniques
- Display of run-time errors directly in the code
- MISRA C 2004 and JSF++ coding violation detection, with direct links to the source file
- Integration with the Eclipse and Microsoft® Visual Studio® IDEs

**P**  
**r**  
**o**  
**v**  
**e**  
**n**

```
static void Pointer_Arithmetic (void)
{
  int array[100];
  int i, *p = array;
  for(i = 0; i < 100; i++, p++)
    *p = 0;

  if(get_bus_status() > 0) {
    if (get_oil_pressure() > 0)
      *p = 5;
    else
      i++;
  }

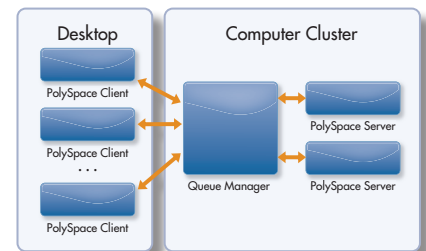
  i = get_bus_status();
  if (i >= 0) { *(p-i) = 10; }

  if ((0 < i) && (i <= 100)) {
    p = p - i;
    *p = 5;
  }
}
```

Green: reliable  
Red: faulty  
Gray: dead  
Orange: unproven

PolySpace Viewer, showing color-coding for each file, procedure, and line of C/C++ code.

Code verification workflow with PolySpace Client for C/C++ and PolySpace Server for C/C++. The queue manager receives the PolySpace verification request and selects the first available server to run the job.



## Errors Detected

Overflows, underflows, division by zero, and other arithmetic errors

Out-of-bounds array access and illegally dereferenced pointers

Read access to noninitialized data

Dead code

Access to null `this` pointer (C++)

Dynamic errors related to object programming, inheritance, and exception handling (C++)

Noninitialized class members (C++)

Other errors, including dangerous type conversions

PolySpace Client for C/C++ supports multiple workflows. For example, you can:

- Submit multiple verification jobs to PolySpace® servers running on a computer cluster or compute farm
- Use one client to verify source code files or classes on a desktop computer
- Use one server with multiple clients, enabling several developers to view and analyze verification results at the same time

### Reviewing Results

PolySpace Client for C/C++ uses color-coding to indicate the status of each element in the code, as follows:

**Green:** proven free of run-time errors

**Red:** proven faulty each time the operation is executed

**Gray:** proven unreachable (may indicate a functional issue)

**Orange:** unproven

MISRA C 2004 and JSF++ coding violations are also detected. The client generates metrics and produces reports that you can use to monitor and help improve code reliability and quality.

You can control the way code verification results are analyzed based on stage in the development process, criticality of the software component, and code certification requirements. PolySpace Client for C/C++ provides both predefined and customizable filters that let you analyze verification results in an iterative process. It optionally tests code marked orange, so you can choose whether to manually review verification results and exercise the unproven code sections with generated test cases.

By confirming the absence of run-time errors and measuring the rate of improvement in code quality, PolySpace Client for C/C++ enables developers, testers, and project managers to target, deliver, or assess code that is free of run-time errors.

### Verifying Program Dynamics

Traditional bug-finding tools either fail to report software errors (yield false negatives) or produce too many warnings (yield false positives). Dynamic testing, which typically uses a finite number of test cases, may miss errors entirely. PolySpace Client for C/C++ verifies all conditions of program execution, for each instruction, taking into account all possible values of every variable at every point in the code. The results provide a formal diagnostic for each operation in the code.

You can use PolySpace Client for C/C++ to support all critical activities in a software development workflow, including:

- Verifying software component integrity and quality under normal and abnormal usage conditions
- Monitoring code quality trends
- Finding and correcting errors during the coding process before test
- Proving the absence of run-time errors in the software component

### Required Products

**PolySpace Server™ for C/C++**

### Related Products

**PolySpace Client™ for Ada**

**PolySpace Model Link™ SL** (for Simulink®)

**PolySpace Model Link™ TL**  
(for dSPACE® TargetLink®)

**PolySpace Server™ for Ada**

**PolySpace UML Link™ RH**  
(for Telelogic® Rhapsody®)

### Platform and System Requirements

For platform and system requirements, visit

[www.mathworks.com/products/polyspaceclient](http://www.mathworks.com/products/polyspaceclient). ■

### Learn More

[www.mathworks.com/products/polyspaceclient](http://www.mathworks.com/products/polyspaceclient)

### Resources

#### VISIT

[www.mathworks.com](http://www.mathworks.com)

#### TECHNICAL SUPPORT

[www.mathworks.com/support](http://www.mathworks.com/support)

#### ONLINE USER COMMUNITY

[www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)

#### DEMOS

[www.mathworks.com/demos](http://www.mathworks.com/demos)

#### TRAINING SERVICES

[www.mathworks.com/training](http://www.mathworks.com/training)

#### THIRD-PARTY PRODUCTS AND SERVICES

[www.mathworks.com/connections](http://www.mathworks.com/connections)

#### WORLDWIDE CONTACTS

[www.mathworks.com/contact](http://www.mathworks.com/contact)

#### E-MAIL

[info@mathworks.com](mailto:info@mathworks.com)