Navigate to Root Cause of Defect

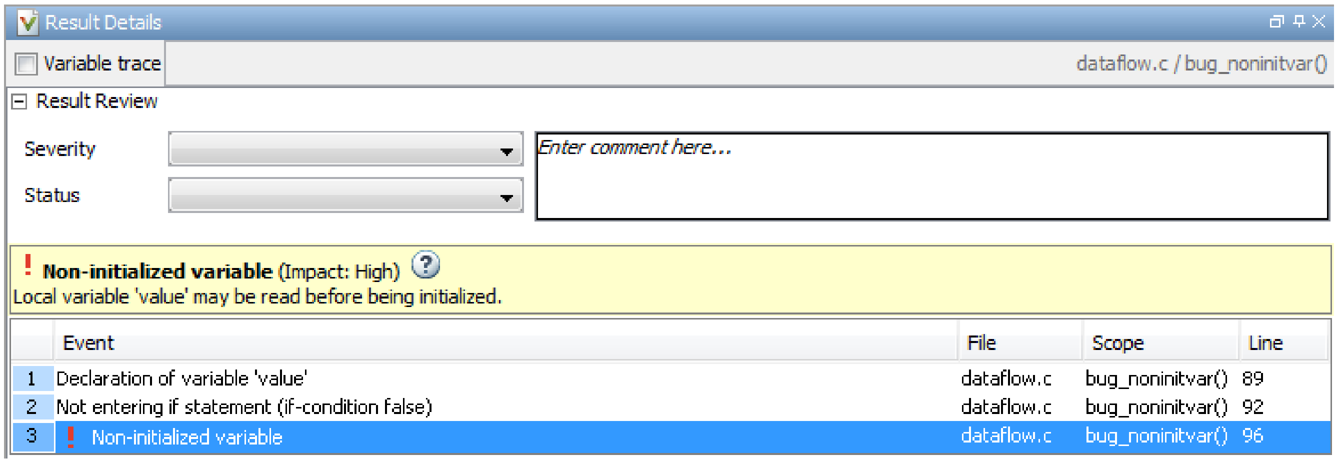
Through the Polyspace® Bug Finder™ user interface, you can navigate to the root cause of a defect in your source code. If you select a result on the **Results List** pane, you see the immediate location of the defect on the **Source** pane. However, the defect can be related to previous statements in your source code.

For instance, a **Non-initialized variable** defect appears at the location where you read a noninitialized variable. However, it is possible that you initialized the variable previously. For instance, the initialization occurred in a branch of a previous if statement and the variable is noninitialized only if that branch is not entered.

Follow Code Sequence Causing Defect

Often, the **Result Details** pane shows the events related to the defect. To see the code statement that the event describes, click the event.

For instance, if an **Array Access Out of Bounds** error occurs in a loop, the **Result Details** pane shows updates to the array index that occur inside the loop. The update statements might physically occur in your code before or after the array access, but because the statements occur in a loop, they are related to the array access.



dataflow.c / bug_noninitvar()

Non-initialized variable (Impact: High)
Local variable 'value' may be read before being initialized.

	Event	File	Scope	Line
1	Declaration of variable 'value'	dataflow.c	bug_noninitvar()	89
2	Not entering if statement (if-condition false)	dataflow.c	bug_noninitvar()	92
3	! Non-initialized variable	dataflow.c	bug_noninitvar()	96

On the **Source** pane, the statements are highlighted in blue and the corresponding line numbers outlined in boxes.

On the **Result Details** pane, you can select the **Variable trace** box, if available. The event sequence expands to show more events related to the defect. The statements that the additional events describe are highlighted in light blue on the **Source** pane.

Navigate to Identifier Definition

Often, to diagnose a defect, you have to navigate to an identifier definition. On the **Source** pane, right-click the identifier name. Select **Go To Definition.**

For instance, the C++ defect **Object slicing** appears at the location where you pass a derived class object by value to a function. The function expects a base class object as parameter. To diagnose this defect, you can navigate to the base and derived class definitions.

To navigate to the derived class definition starting from the defect location:

1. Right-click the derived class object name and select **Go To Definition.**
2. In the derived class object definition, right-click the derived class name and select **Go To Definition.**

If a definition is not available to Polyspace, selecting the option takes you to the declaration. For instance, Polyspace Bug Finder displays results in real time as they are produced. The software displays results on some files while others are not yet analyzed. In your results, if you select a function and then select **Go To Definition**, and the function definition is not yet analyzed, selecting the option takes you to the function declaration.

Navigate to Identifier References

Often, to diagnose a defect, you have to see the locations where an identifier is used.

For instance, an if statement shows the **Dead code** defect. You want to understand why the variable that controls entry to the if statement has a certain set of values. Therefore, you want to see previous assignments to that variable.

To navigate to previous locations where an identifier is used:

1. Right-click the identifier name and select **Search For All References.**
   The search results appear on the **Search** pane with the current location highlighted.
2. Click each search result, starting backward from the highlighted result.
3. The option **Search for All References** is not available in some cases. For instance, if you right-click a C++ virtual function, this option is not available.
   Use one of the following options to search for occurrences of the identifier name:
   - **Search For `Identifier _ name` in Current Source File**
   - **Search For `Identifier _ name` in All Source Files**
4. If reviewing a defect requires deeper navigation in your source code, you can create a duplicate source code window that focuses on the defect while you navigate in the original source code window.
   a. Right-click on the **Source** pane and select **Create Duplicate Code Window.**
   b. Right-click on the tab showing the duplicate file name and select **New Vertical Group.**
   c. Perform the navigation steps in the original file window while the defect still appears on the duplicate file window.
   d. After reviewing the defect, click the button on the **Results List** pane to return to the defect location in the original file window. Close the duplicate window.