Topaz Workbench Program Analysis and Runtime Visualizer User Guide

Table of Contents

Program Analysis

Welcome to Program Analysis

Topaz Workbench Program Analysis is an Eclipse plug-in you can use to create, view, manipulate, export, and print graphical representations of COBOL and PL/I program structure. The logic flow of each procedure or paragraph and the data flow for a selected field can also be viewed. The hierarchical relationships among copybooks and programs in a Compuware mainframe project can also be shown.

The Program Analysis plug-in includes the **Program Analysis** perspective for working with program structure, logic flow, data flow, and program and copybook hierarchies.

Program analysis is initiated from a SlickEdit session and opens a chart of the program's structure. A logic flow chart for a selected procedure can then be invoked from that program structure chart. A data flow chart can be invoked by clicking a field in SlickEdit or entering it in a text box. Another chart filters out complexity by presenting only the selected structure chart node, its callers (parents), and callees (children). Analyzing an online mainframe project in Project Explorer enables you to view that project's hierarchy of program calls and copybook references.

Related Topics

Getting Started with Program Analysis

Getting Started with Program Analysis

This topic will help you prepare to start using Program Analysis.

To use Program Analysis for mainframe data, you must define an HCI host connection and connect Topaz® Workbench or Eclipse/RDz to a mainframe system. To define your connection, click **Window > Preferences**. Then expand **Compuware > Host Connections** and add an HCI host connection before continuing.

A host connection between the mainframe host and Topaz should have already been defined during the installation and configuration of Topaz Workbench. If you do not have host connection information (typically a port number and a host name), contact your system administrator to obtain specific connection configurations for your site for establishing Host Connections. (For system administrators, see the *Topaz Workbench Installation Guide* for additional information on installing Topaz Workbench.)

To enable Program Analysis, you must also set the URL for Compuware Enterprise Services. For more information, see the *Topaz Workbench User Guide*.

Note: When using File-AID Services in support of File-AID Data Privacy in Topaz Workbench, Host connections are centrally configured on the server and available to all users that define File-AID Services in their Compuware preferences.

Usage Restrictions

All Languages

- Program analysis does not support DBCS, the pound sterling character in 1004, Desktop Publishing code page or 852, Latin Multi-Lingual code page.

COBOL

- Nested programs are not supported. Only the first program is analyzed.
- Report Writer syntax is not supported.
- Object oriented COBOL is not supported, only tolerated.
- Program analysis does not support the omission of the PROCEDURE DIVISION header or label.

PL/I

- PL/I programs containing preprocessor directives other than the following must have the directives resolved before they can be analyzed:
  - %INCLUDE
  - %PRINT
  - %NOPRINT
  - %SKIP
  - %PROCESS
- The following are supported:
  - %IF/%ELSE statements
  - %DO blocks (including loops)
  - %INCLUDE
  - Assignment statements
  - %ACTIVATE / %DEACTIVATE
  - %DECLARE
  - Procedures that return a value
  - GOTO (within procedures).
- The following built-in functions are supported:
  - NOTE (within procedures)
  - LENGTH
  - INDEX
  - SUBSTR
  - PARMSET (within procedures).
- Partial support is provided for the following:
  - Typed variables. (In many cases variables are treated as strings and converted to numbers as needed.)
  - Procedure calls using named parameters.
- The following are ***not*** supported:
- Use of compile options RULES(LAXPUNC), BLANK, and NAMES
- Alternate Include processor IDs specified via the PP(INCLUDE(ID(“..”)) compile option
- Multiple level zero procedures (Likewise, statements outside the level zero procedure are ignored.)
- %ACTIVATE / %DEACTIVATE for procedures (Once defined a procedure is always active.)
- Calls to procedures without specifying parenthesis (no parameters)
- Calls made to a procedure before that procedure is declared.
- The EBCDIC/ASCII host code page should be used when downloading or file transferring PL/I programs; otherwise, some PL/I-specific characters may be translated incorrectly.
- Every statement within a PL/I program must be terminated by a semicolon.
- Label variables are not supported on GOTO statements, only tolerated.

Related Topics

Opening Program Analysis

Using Program Analysis

Opening Program Analysis

Following are the instructions for opening Program Analysis.

To open Program Analysis

1. Start Topaz Workbench.

2. Use Host Explorer to open the desired program in SlickEdit and optionally download any associated copybooks. If the **Program Analysis** perspective is not already selected, do either of the following:

   - Right-click the program source and select **Perform Program Analysis**.

- From the **Compuware** menu, select the **Program Analysis** perspective.

Program Analysis creates a chart of the program's structure and displays it in a separate view. Each program structure chart you open is displayed in a new view labeled with the program name. If the program cannot be analyzed for some reason, an error message will appear.

Related Topics

Using Program Analysis

Working with the Program Structure Chart

Using Program Analysis

Using Program Analysis

The Topaz Workbench **Program Analysis** perspective is used to present information about a program in chart form with related problem and metrics views. Components of the **Program Analysis** perspective include:

- Program structure chart
- Logic flow chart
- Data flow chart and table
- Compiler Options view
- Callers/Callees chart
- Problems view
- Program Summary view
- Procedures view
- Properties view.

The remaining topics in this section explain how to work with the various charts and views.

Related Topics

Working with the Program Structure Chart

Working with the Logic Flow Chart

Working with the Data Flow Chart

Working with the Compiler Options View

Working with the Callers/Callees Chart

Viewing Code Problems

Viewing the Program Summary

Viewing Procedure Metrics

Working with the Program Structure Chart

The **Program Analysis** perspective includes a ***program structure chart*** that presents the program as a collection of paragraph or procedure nodes, rounded rectangular boxes that display the name and line number of the paragraph or procedure. These nodes are connected by lines representing the flow of data within the program, with arrowheads indicating the direction of data flow: For an input operation, such as READ or SQL FETCH, the arrowhead points from the data object to the calling object. For output operations, such as WRITE or SQL INSERT, the arrowhead points from the calling object to the data object. No arrowhead is shown if the I/O is neither clearly input or output, such an OPEN, CLOSE, or START.

Chart Characteristics

Symbols in each node indicate the type of program resource as follows:

E	indicates the program's entry point
P	indicates a paragraph or procedure
C	indicates a program call
DD	indicates a DD (data definition)
D	indicates a DB2 table
D	indicates dead code

The tab title for the program structure chart displays the program name and the characteristic indicated by the chart’s colored highlighting. Paragraph or procedure nodes in the program structure chart have different colors based on the selected metric color. Right-clicking on the chart and selecting **Metric Color** displays a menu of available node color choices with their corresponding program characteristics:

□	None
🟩	McCabe
🟩	Statements
🟦	Blocks
🟦	Conditionals
🟦	GOTOs
🟪	File I/O
🟪	Performs
🟪	Program Calls
🟥	Starting Line.
🟥	SQL Statements

When nodes are colored in shades of green indicating their McCabe Complexity metric, the higher the number, the darker the shading relative to the other nodes in the chart. The darkest shading indicates the most complex parts of a given program, and the lightest indicates the least complex, even though the underlying numbers will vary for different programs.

Other metric colors are similarly lighter or darker based on the relative number of statements, blocks, conditionals, GOTOs, performs, starting lines, or SQL statements present in those paragraph or procedure nodes. If File I/O is selected for metric color, only the paragraph or procedure nodes where file input/output occurs are colored.

Interacting with the Chart

Clicking on a node in the program structure chart selects and highlights it with a bold border around the node, while displaying the selected paragraph or procedure in the logic flow chart. Any paragraphs or procedures that call and/or are called by the selected paragraph or procedure are shown in the **Callers/Callees** chart, and the source is synchronized to the corresponding statement in SlickEdit. Ctrl+clicking one or more additional nodes selects and highlights them, while also highlighting the logical path between the selected nodes. Hovering over an execution line displays the names of the two connected nodes.

Each node in the program structure chart can be moved to a new location by dragging it with the left mouse button pressed. The contents of the entire chart can also be moved either by using the scroll bars or by placing the cursor on an open area then dragging with the left mouse button pressed.

Nodes can be arranged top to bottom (the default), left to right, or bottom to top by right-clicking anywhere in the program structure chart and selecting **Layout > Top Down**, **Layout > Left to Right**, or **Layout > Bottom Up**.

To make the program structure chart easier to view, the content that appears in each of the nodes can be controlled by clicking the **Toggle Text** toolbar button and selecting **No Text**, **Line Number**, or **Full Text**. If you select **No Text** (the default), only the symbol associated with the corresponding paragraph or procedure is shown in each node. If you select **Line Number**, the symbol and the source code line number are shown in each node. Select **Full Text** to display the symbol, the line number, and the name of the paragraph or procedure in each of the nodes.

To refresh the program structure chart from the SlickEdit source, either click the **Reanalyze Program** toolbar button, or right-click the source and select **Reanalyze Program**. The program structure chart is also refreshed when you save your program.

Collapsing and Expanding Nodes

Program structure chart nodes can be collapsed to hide subsequent nodes by right-clicking and selecting **Collapse**. The collapsed node includes a red indicator in the lower right corner showing the number of levels that have been collapsed. To redisplay nodes hidden under a collapsed node, right-click the node and select **Expand**, **Expand All**, or **Expand Levels**. Selecting **Expand** expands the collapsed node one level. Selecting **Expand All** fully expands all nodes under the collapsed node. **Expand Levels** provides sub-menu choices to expand the collapsed node by 3, 5, or 7 levels. If the node has been collapsed more levels than it is expanded, subsequent nodes that remain collapsed include a red indicator showing the number of levels that could still be expanded. For easier viewing after the desired nodes have been collapsed, the chart can be redrawn without the hidden nodes by right-clicking anywhere in the program structure chart and selecting the desired **Layout>** option.

Viewing a Subsection of the Chart

Large program structure charts can be difficult to view and navigate. To make it easier to focus on a particular section of the chart, right-click the node you are interested in and select **View Subgraph > Make Root**. The selected node becomes the root node for a program structure sub-chart, and an indicator appears in the top right corner of the node showing how many parent nodes directly above have been hidden. A node must have children to be set as root. To display one level higher, right-click the new root node and select **View Subgraph > Show Parents**. To redisplay the full program structure chart, right-click anywhere in the chart and select **View Subgraph > Show All Nodes**.

Zooming and Spacing

Each chart in the **Program Analysis** perspective can be zoomed in and out individually, and the spacing between nodes can be increased and decreased, independent of the zoom level. The zoom percentage drop-down at the top of the each chart includes a number of preset zooms from 5% to 400%, along with the choices **Page** to fit the chart fully within the view, **Height** to fit the chart to the height of the view, and **Width** to fit the chart to the width of the view. Ctrl+scroll can also be used to quickly zoom in or out through the preset zoom values. A chart can also be zoomed in small increments by Alt-clicking for larger or Shift-Alt-clicking for smaller. The default is 100%.

To view details about an individual node in full size while zoomed out smaller than 100%, simply hover over the desired node. A full-size tooltip showing the associated line of code is temporarily displayed.

The numeric spinner next to the zoom level controls the spacing between nodes, with higher numbers providing more space. Available spacing values are 0 through 50. The default is 0.

Charting of Dead Code

If "dead code" that can never be executed exists in the program, the corresponding nodes appear separated from the main portion of the program structure chart and are indicated by a D and gray shading. The nodes and code blocks with dead code can be toggled between hidden and visible (the default) in both the program structure chart and the related logic flow chart at once by clicking the **Hide Dead Code** toolbar button. The display of dead code in other open charts remains unaffected until one of those charts is selected. Dead code in a program structure chart is associated with code problem I006 (Unentered procedure), and dead code in a logic flow chart is associated with code problem I005 (Inexecuteable statement). See Code Problem Descriptions for more information.

Related Topics

Working with the Logic Flow Chart

Working with the Data Flow Chart

Working with the Callers/Callees Chart

Viewing Code Problems

Working with the Logic Flow Chart

The **Program Analysis** perspective also includes a ***logic flow chart*** that shows the flow of the code in whatever paragraph or procedure has been selected in the program structure chart. That selected paragraph or procedure is presented as a collection of code blocks, rounded rectangular boxes representing a sequential group of statements in the program. These nodes are connected by lines representing their logical sequence within the program, with arrowheads indicating the direction of the execution path. Symbols in each node indicate the type of program resource as follows:

P	indicates a paragraph or procedure's entry point
C	indicates a program call
DD	indicates a DD (data definition)
D	indicates a DB2 table

Each code block in a logic flow chart has one entry point and one exit point. The entry point is indicated visually with a P and a bold border around the node, and the contents of the entry point is shown at the top of the view. Each node displays the line number and action associated with that node, and hovering over the node temporarily displays the associated line of code. If the code block represented is an invisible statement (an internally generated statement), the line number shown is replaced by ellipses (.....). This is the case with generated scope terminators, periods, and label-returns.

Clicking a logic flow chart node selects and highlights it with a bold border while synchronizing to the corresponding statement in SlickEdit. Ctrl+clicking one or more additional nodes selects and highlights them while also highlighting the logical path between the selected nodes.

Note: To make the logic flow chart easier to navigate, not all statements are shown. They are grouped into code blocks, and only the first statement in the block is presented. In each code block, if the first statement is executed, all following statements will be executed; there is no branching. This allows for a compressed and easier to navigate chart.

Nodes can be arranged top to bottom (the default), left to right, or bottom to top by right-clicking anywhere in the logic flow chart and selecting **Layout > Top Down**, **Layout > Left to Right**, or **Layout > Bottom Up**.

Logic flow chart nodes can be collapsed to hide subsequent nodes by right-clicking and selecting **Collapse**. The collapsed node includes a red indicator in the lower right corner showing the number of levels that have been collapsed. To redisplay nodes hidden under a collapsed node, right-click the node and select **Expand**, **Expand All**, or **Expand Levels**. Selecting **Expand** expands the collapsed node one level. Selecting **Expand All** fully expands all nodes under the collapsed node. **Expand Levels** provides sub-menu choices to expand the collapsed node by 3, 5, or 7 levels. If the node has been collapsed more levels than it is expanded, subsequent nodes that remain collapsed include a red indicator showing the number of levels that could still be expanded. For easier viewing after the desired nodes have been collapsed, the chart can be redrawn without the hidden nodes by right-clicking anywhere in the logic flow chart and selecting the desired **Layout>** option.

If "dead code" that can never be executed exists in the program, the corresponding nodes appear separated from the main portion of the logic flow chart and are indicated by a Ⓓ and gray shading. The nodes with dead code can be toggled between hidden and visible (the default) in both the logic flow chart and the related program structure chart at once by clicking the toolbar button. The display of dead code in other open charts remains unaffected until one of those charts is selected. Dead code in a logic flow chart is associated with code problem I005 (Inexecuteable statement). See Code Problem Descriptions for more information.

Preferences can be used to customize node size and line wrapping of node content in the logic flow chart. To change your preferences, click **Window>Preferences**. The **Preferences** dialog box appears. Expand **Compuware** and select **Data Visualizer**. The Preferences for the Data Visualizer, which also control the logic flow and data flow charts, are shown. See Specifying Preferences.

Related Topics

Working with the Program Structure Chart

Working with the Data Flow Chart

Working with the Data Flow Chart

The flow of data within a program is shown graphically and in table form in the ***data flow chart*** view. The **Chart** tab and **Table** tab at the bottom of the view are used to select the desired representation. Clicking on a variable field or a referenced literal in SlickEdit, or specifying a variable in the view’s drop-down text box, will chart the flow of data associated with that variable. On the **Chart** tab, fields are shown as nodes, rounded rectangular boxes representing data items. These nodes are connected by lines representing the flow of data within the program, with arrowheads indicating the direction of the data flow. The node for the selected variable is highlighted in the data flow chart with a red border.

The lines connecting data flow chart nodes represent the type of data flow:

- Solid lines indicate moves.
- Dashed lines indicate conditional flow.
- Dotted lines indicate an alias.
- Dash-dot-dash lines indicate a conditional flow alias.

Curved line are used to reduce overlapping.

The data flow chart includes six buttons that control what is shown in the chart. If a button is depressed, that type of data flow is shown, and if it is not depressed, that type of data flow is hidden. These buttons can be used to tailor the contents of the data flow chart as follows:

0..n	Show All Depths. By default, only data flows with a **Depth** of 0 (those directly related to the selected variable) are shown. Depress to show data flows for all levels of **Depth**.
	Show Aliases. Depress to show data flow for aliases of the selected variable. Default is not depressed.

?=?	Show Conditional Flow. Depress to show data flow — either to or from the variable — that occurs as a result of conditional logic. Default is depressed.
	Show Flow To. Depress to show the data flowing to the selected variable. Default is depressed.
	Show Flow From. Depress to show the data flowing from the selected variable. Default is depressed.
	Pin Data Flow. Toggles whether the view is linked to the active editor. When selected, the current data flow chart view is retained. Values shown in the corresponding **Properties** view are also retained. When deselected, the views are synchronized with selections made in the active editor. Default is deselected.

The text box at the top of both the **Chart** tab and **Table** tab can be used to enter the name of the desired variable. As characters are entered, a list of matching variables is displayed. Click the arrow on the right end of the text box to display a list of all variables for which data flow can be displayed.

Clicking a data flow chart node selects and highlights it with a bold border and synchronizes the SlickEdit view to where the variable was defined in the program. The characteristics of the data flow chart node are displayed on the **Table** tab and in the **Properties** view. Ctrl-clicking one or more additional nodes selects and highlights them while also highlighting the data flow path between the selected nodes.

Nodes can be arranged top to bottom (the default), left to right, or bottom to top by right-clicking anywhere in the data flow chart and selecting **Layout > Top Down**, **Layout > Left to Right**, or **Layout > Bottom Up**.

Data flow chart nodes can be collapsed to hide subsequent nodes by right-clicking and selecting **Collapse**. The collapsed node includes a red indicator in the lower right corner showing the number of levels that have been collapsed. To redisplay nodes hidden under a collapsed node, right-click the node and select **Expand**, **Expand All**, or **Expand Levels**. Selecting **Expand** expands the collapsed node one level. Selecting **Expand All** fully expands all nodes under the collapsed node. **Expand Levels** provides sub-menu choices to expand the collapsed node by 3, 5, or 7 levels. If the node has been collapsed more levels than it is expanded, subsequent nodes that remain collapsed include a red indicator showing the number of levels that could still be expanded. For easier viewing after the desired nodes have been collapsed, the chart can be redrawn without the hidden nodes by right-clicking anywhere in the data flow chart and selecting the desired **Layout>** option.

The **Table** tab can be used to view the contents of the **Chart** tab in a table layout. Available information for each data item in the view is provided in the following columns:

- Data Item
- Defined
- Line
- Data Action
- Verb
- Depth
- Definition member
- Definition member dataset
- Length (Physical)
- Length (Logical)
- Picture
- Usage
- Memory offset
- Parent
- Level
- Initial value
- Occurs

- Where modified
- Where used.

The values in the **Table** tab can be exported for later comparison following code changes. For more information, see Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views.

Preferences can be used to customize node size and line wrapping of node content in the data flow chart. To change your preferences, click **Window > Preferences**. The **Preferences** dialog box appears. Expand **Compuware** and select **Data Visualizer**. The Preferences for the Data Visualizer, which also control the logic flow and data flow charts, are shown. See Specifying Preferences.

Related Topics

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

Working with the Program Structure Chart

Working with the Logic Flow Chart

Working with the Compiler Options View

The Topaz Workbench **Program Analysis** perspective includes a **Compiler Options** view that allows you to view and modify the compile options for a source member open in SlickEdit. The **Compiler Options** view, which is adjacent to the **Data Flow** view, indicates the language, Enterprise COBOL or PL/I, of the source member and allows you to change certain parameters.

Specifying Compiler Options for a program member

1. Open a program member from Host Explorer.
2. To specify COBOL options, first select **Enterprise COBOL** from the **Language level** dropdown list, then specify desired COBOL values for **SQL Apostrophe Quote** and **COBOL Apostrophe Quote**.
3. To specify PL/I options, first select **PL/I** from the **Language level** dropdown list, then specify desired PL/I values for **Left Margin**, **Right Margin**, **NOT Symbol**, **OR Symbol**, and **SQL Apostrophe Quote**. To optionally use PL/I preprocessor values:

a. Select the **Enable PL/I Preprocessor** checkbox.

b. To specify preprocessor parameters, click the **PL/I Preprocessor Parameters Edit** button.

c. To add a variable/value pair, click **New**, specify a **Name** and **Value** in the **Change Preprocessor Parameter** dialog box, then click **OK**.

d. To modify an existing variable/value pair, select it, click **Edit**, modify the **Name** and/or **Value**, then click **OK**.

e. To delete an existing variable/value pair, select it and click **Remove**.

f. When the entries in the **PL/I Preprocessor Parameters** dialog box are configured as desired, click **OK**.

4. Click **Apply** to accept your changes.

Working with the Callers/Callees Chart

The **Program Analysis** perspective includes a ***Callers/Callees chart*** that shows only the node selected in the program structure chart along with its caller (parent) and callee (children) nodes. As in the program structure chart, paragraph or procedure nodes are shown as rounded rectangular boxes that display the name of the paragraph or procedure and are connected by lines representing the flow of data within the program, with arrowheads indicating the direction of data flow: For an input operation, such as READ or SQL FETCH, the arrowhead points from the data object to the calling object. For output operations, such as WRITE or SQL INSERT, the arrowhead points from the calling object to the data object. No arrowhead is shown if the I/O is neither clearly input or output, such an OPEN, CLOSE, or START.

Chart Characteristics

Symbols in each node indicate the type of program resource as follows:

E	indicates the program's entry point
P	indicates a paragraph or procedure
C	indicates a program call
DD	indicates a DD (data definition)
D	indicates a DB2 table
D	indicates dead code

Interacting with the Chart

Clicking any of the nodes in the Callers/Callees chart will synchronize to the corresponding source statement in SlickEdit.

Note: Clicking a node in the Callers/Callees chart does ***not*** automatically update the program structure chart.

Right-click on a Callers/Callees node and select **Show Callers/Callees** to update both the program structure chart and the Caller/Callees chart while synchronizing to the corresponding source statement in SlickEdit.

The content of the Callers/Callees chart can be updated by selecting a different program structure chart node or by clicking anywhere in the source displayed in SlickEdit.

Related Topics

Working with the Program Structure Chart

Viewing Code Problems

When you open a program in SlickEdit and the **Program Analysis** perspective, the program is parsed and analyzed to build the program structure chart and provide in-depth information about the program. The code problems noted during the analysis, including those that could cause confusion or unintended results, are listed in the Eclipse **Problems** view. These are not the same as compiler diagnostics and do not replace them. If you wish to see the compiler diagnostics for your program see Submitting Compiles and Displaying Compile Diagnostics.

Note: The **Problems** view, if not visible, can be opened from the **Window** menu by selecting **Show View > Other > General > Problems**.

Note: To ensure that Info problems are displayed, click the **Problems** view menu button ▽ and select **Show > Show All**.

To view the description of a code problem, click the entry in the **Problems** view, then click the ? button. The description appears as a context-sensitive Help item. To view the description of a different problem, simply click it in the **Problems** view. To view an overall description of code problems, click the heading for any of the listed problem types, then click the ? button. A list of all code problems is provided in Code Problem Descriptions.

To go directly to the line of code associated with a code problem in SlickEdit, double-click the entry in the **Problems** view.

Each code problem identified is also indicated with a colored mark in the overview ruler adjacent to the scroll bar in SlickEdit. This allows you to easily see where the greatest concentration of code problems are in the program. Click an overview ruler mark to go directly to the line containing the code problem.

Note: When numerous errors exist, the overview ruler marks for Info problems can overlay the marks for Error and Warning problems. To ensure the marks for Error and Warning problems are not overlaid by marks for Info problems, you may choose to turn off Info marks. Click **Window > Preferences > General > Editors > Text Editors > Annotations**. Select **Info** from the **Annotation types** list, then in the **Show in** section uncheck the **Overview ruler** checkbox. Click **OK** to save your preferences.

Each code problem is also indicated by a marker in the vertical ruler to the left of the SlickEdit source display. Hover over a marker to show a short description of the problem, including the path to the affected program.

The values in the **Problems** view can be exported for later comparison following code changes. For more information, see Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views.

Related Topics

Code Problem Descriptions

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

Code Problem Descriptions

This topic lists and provides descriptions for the various code problems that can be identified in the **Program Analysis** perspective.

Severe Messages

S001 IDENTIFICATION DIVISION header not found.

The IDENTIFICATION DIVISION header is missing. Be sure that you have entered a valid COBOL program.

S002 PROGRAM-ID not found.

The PROGRAM-ID is missing. Be sure that you have entered a valid COBOL program with a valid PROGRAM-ID.

S003 PROGRAM-ID paragraph not found.

The PROGRAM-ID paragraph is missing. Be sure that you have entered a valid COBOL program with a valid PROGRAM-ID paragraph.

S004 Stack overflow. The token look-ahead exceeded nnnn.

The analysis encountered a stack overflow. Be sure that you have entered a valid COBOL program.

S005 Unable to open input file.

The analysis could not open the input file.

S006 Preprocessor DLL returned invalid return code: xxxx.

The user exit returned an invalid return code, where rc is the return code. Parsing and analysis terminates. The return codes listed earlier are the only valid ones.

S007 Level 0 procedure not found.

A PL/I Procedure statement was not found.

S008 Ambiguity detected. Internal parser error.

There was an error in the analysis because there are multiple rules that match the source it is parsing and it is ambiguous which rule to follow. Contact Compuware Customer support.

Error Messages

E001 A REPLACE clause is missing a BY or a period.

Analysis encountered a REPLACE clause without a BY phrase or period. Enter the BY phrase or period.

E002 Procedure name xxxx is undeclared.

A procedure name is referenced by a statement, but the procedure name does not exist in the input program. Analysis stops and the structure chart and logic flow graph are not generated. Add the procedure name.

E005 Recursive Copy/Include.

A nested copy member is opened twice in the program. Correct the program to stop the COPY/INCLUDE recursion.

E006 Found xxxx, but expected yyyy or yyyy or yyyy or ....

Program Analysis received unexpected data. If the instances of unexpected data cause the message to exceed 128 bytes, the yyyy series is replaced by ellipsis (...). Analysis stops and the structure chart and logic flow graph are not generated.

E007 Unable to open SQL INCLUDE member xxxx.

Analysis was unable to open the SQLCA or SQLDA INCLUDE member. We should be able to successfully analyze your program. If you would like to have this member included, check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E008 Illegal qualifier for xxxx.

Analysis encountered an illegal qualifier. Enter a valid qualifier.

E009 Invalid sequence of characters.

Analysis encountered an invalid sequence of characters. Enter a valid character sequence.

E010 Missing COPY/INCLUDE member name.

A member name is missing from a COPY or %INCLUDE statement. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E011 Object of REDEFINES not found.

Analysis could not find the object of a redefine phrase. Check the redefine phrase.

E012 Continuation character expected. End of literal assumed.

A quoted string was not properly terminated. Enter the proper termination.

E013 xx must begin in Area B. Proceeding as if xx began in Area B.

Analysis encountered a statement that begins in the incorrect column.

E016 Unable to open COPY/INCLUDE member xxxx.

Analysis was unable to open the COPY/INCLUDE file. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E017 Unable to open SQL INCLUDE member xxxx.

Analysis was unable to open the SQL INCLUDE member. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E018 Unable to open ++INCLUDE member xxxx.

Analysis was unable to open the ++INCLUDE member. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E019 Unable to open -INC member xxxx.

Analysis was unable to open the -INC member. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E020 Unmatched exception phrase.

Analysis encountered an exception phrase without a matching statement. Analysis stops and the structure chart and logic flow graph are not generated. Delete the phrase or enter a matching statement.

E021 Unmatched ELSE phrase.

Analysis encountered an ELSE phrase without a matching IF statement. Analysis stops and the structure chart and logic flow graph are not generated. Delete the phrase or enter a matching IF statement.

E022 Unmatched WHEN phrase.

Analysis encountered a WHEN phrase without a matching statement. Analysis stops and the structure chart and logic flow graph are not generated. Delete the phrase or enter a matching statement.

E023 xxxx is an illegal character.

Analysis encountered a character that is not in the COBOL character set. Analysis stops and the structure chart and logic flow graph are not generated. Check the COBOL dialect and settings in the Lang Level tab in the Settings dialog box. Enter the proper characters.

E024 xxxx is not valid in the PICTURE clause.

Analysis encountered data that is not permitted in a PICTURE clause. Re-enter the PICTURE clause with valid data.

E025 xxxx is undefined.

Analysis encountered an undefined field. Check the spelling of the field name. If the spelling is correct, make sure the field name is not in a missing copybook or %INCLUDE. Then check settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

E026 Unmatched AT END phrase.

Analysis encountered an unmatched AT END phrase. Analysis stops and the structure chart and logic flow graph are not generated. Check the logic before and after the AT END phrase.

E027 FILE-CONTROL paragraph SELECT clause not found for FD and 01-level record.

Analysis could not find a FILE-CONTROL paragraph SELECT clause for an FD and 01 record. Add the SELECT clause.

E028 FILE-CONTROL paragraph SELECT clause not found for FD.

Analysis could not find a FILE-CONTROL paragraph SELECT clause for an FD. Add the SELECT clause.

E029 END label xxxx is not a label on any open block.

A labeled END statement could not find the statement that created the block.

E030 Scope terminator found without a matching statement.

Analysis encountered a scope terminator without a matching statement. This could be caused by having a period at the end of a statement before reaching the END-IF.

E031 Nested programs are not supported and will be excluded from analysis.

The analysis will be confined to the main program, and nested programs will be excluded.

Warning Messages

W001 A period is required.

A required period is missing. Add the period.

W003 Missing period for COPY statement.

The COPY statement requires a period. Add the period.

W004 xxxx is not declared.

The analysis encountered an undeclared SQL data element. Check the file name and spelling as well as settings for copy paths to ensure the copybook download function is properly configured. For more information, see Downloading_Copybooks.

W005 xxxx is not uniquely qualified.

The analysis encountered duplicate qualifiers. Check the qualifiers and enter unique ones.

W006 Missing period in REPLACE statement.

The REPLACE statement is missing a period. Add the period.

W007 Host variable without preceding colon.

The host variable is not preceded by a colon. In some contexts, you can specify a host variable in an SQL statement without a preceding colon. The analysis issues a warning message and processes the statement as if the missing colon is present. Compuware strongly recommends that all host variables be preceded by a colon for the following reasons:

- It enhances readability by making the host variables easier to identify.
- It increases portability, since other varieties of SQL do not let you omit a colon.
- Future versions of DB2, in conformity with ANSI standards, may not allow the omission.

W008 Found x, but expected y.

The analysis either expected an apostrophe (') as a string delimiter but found a quote ("), or expected a quote but found an apostrophe.

W010 Could not load preprocessor dll xxxx.

The Program Analyzer could not load the Filename.dll, the preprocessor specified on the Preprocessor tab of the Settings dialog box. The Program Analyzer runs without trying to invoke the preprocessor DLL. The operating system was unable to load the DLL. Be sure that any run-time support that the DLL needs to execute is available. Also, if the DLL file does not include the full drive:pathname, its directory must be known to the current operating system (via LIBPATH or PATH).

W011 Object oriented COBOL is not supported.

Object oriented COBOL is not supported, only tolerated.

W012 Preprocessor directive will be ignored.

PL/I programs containing preprocessor directives other than %INCLUDE, %PRINT, %NOPRINT, %SKIP, and %PROCESS must have the directives resolved before they can be analyzed. For more information, refer to the Usage Restrictions section in Getting Started with Program Analysis.

W013 All statements outside the level 0 procedure will be ignored.

Statements outside the level 0 procedure were encountered, or more than one level 0 procedure was found.

W014 Label variables are not supported on GOTO statements.

Label variables are not supported on GOTO statements, only tolerated.

W015 Reference to SQL procedure xxxx is ignored.

COBOL code references a paragraph that is generated by the SQL pre-processor. We do not display code that is generated by the SQL pre-processor, therefore these statements are ignored.

W016 Statement contains wrong number of INTO variables.

The number of variables specified in an INTO clause is not the same as the number of select-list elements.

W017 Report Writer Definition found in this program.

Report Writer Entry found in this program. Report Writer syntax is not supported, therefore any Report Writer verbs that might follow will be rejected.

Informational Messages

I001 End of sentence should precede xxxx. Assumed present.

Analysis encountered a procedure label or the input source’s END-OF-FILE without a preceding period. The end of sentence is assumed.

I002 In-line PERFORM found without matching END-PERFORM.

Analysis encountered a PERFORM statement without a corresponding END-PERFORM statement.

I003 Recursive PERFORM.

Analysis encountered a recursive PERFORM code problem. Indicates that the paragraph contains recursion (a PERFORM statement that can reach itself before the PERFORM is finished).

I005 Inexecutable statement.

The program's logic did not permit the analysis to reach a particular statement, so the statement was not analyzed. This means that this statement is “dead code” and would never be executed.

I006 Unentered procedure.

The program's logic did not reach a particular procedure.

I007 Unable to open SQL INCLUDE member xxxx.

Analysis was unable to open the SQLCA or SQLDA INCLUDE member. These files are available in your installation directory. You can add these files to the copy paths on the Copybook Paths tab (for COBOL) or the Include Paths tab (for PL/I) in the Settings dialog box.

I008 PRV : violates range xxxx thru xxxx.

Analysis encountered a GO TO statement that branched outside its Perform Range (xxxx through xxxx). This GO TO statement created a Perform Range Violation (PRV) code problem.

I009 Too many error or warning messages.

You have exceeded the number of messages that can be stored. Analysis will continue, but it will not display any more messages.

I010 SRV: violates scoped xxxx.

Analysis encountered a GO TO or NEXT SENTENCE statement that branched outside the range of the specified (xxxx) scoped delimited statement. This created a Scoped Range Violation (SRV) code problem. In the case of an IF statement it indicates that the GO TO or NEXT SENTENCE referred to will end up past the END-IF.

I011 Unresolved scoped xxxx.

The xxxx scope does not execute to its corresponding end scope statement. This message is usually caused by a GO TO out of a scope or a non-returning paragraph called from a scope.

I012 Runaway logic caused by GO or in PROCEDURE DIVISION.

Analysis detected that the program does not contain a terminate execution statement or program logic bypassed terminate execution statement.

I013 Block(s) not explicitly closed.

A block of code does not have a matching end statement.

I014 File xxxx is not used.

File is not used. This will occur if a file is defined in a program but it is not referenced in a statement other than an open or close.

I015 Table xxxx is not used.

Table is not used. This will occur when a table is declared in a program but it is not used in another SQL statement.

Viewing the Program Summary

When you open a program in SlickEdit and the **Program Analysis** perspective, the program is parsed and analyzed to build the program structure chart and provide in-depth information about the program. The **Program Analysis** perspective includes a **Program Summary** view that provides valuable information about the program being charted. The **Files Defined**, **Files Opened**, and **Files Used** values can be expanded to list the associated DD names. The **DB2 Tables Defined** and **DB2 Tables Used** values can be expanded to list the associated DB2 tables. Differences can help in identifying files and tables that are defined but never used. The **Maintenance Effort** can be expanded to list the associated maintenance effort factors.

Note: The **Program Summary** view, if not visible, can be opened from the **Window** menu by selecting **Show View > Other > Program Analysis > Program Summary**.

The two columns of the **Program Summary** view, **Measurement** and **Value**, contain the following data:

Measurement	**Value**
Lines of Code	The total number of source lines in the program (excluding blank lines).
Comment Lines	The total number of commented lines in the program (excluding lines that contain executable code and a comment).
Statements	The number of executable lines of code in the program.
Files Defined	The number of files that are defined in the program. This field can be expanded to list the DD names defined.
Files Opened	The number of files that are opened in the program. This field can be expanded to list DD names opened.
Files Used	The number of files that are used in a Read or Write statement. This field can be expanded to list DD names used.
DB2 Tables Defined	The number of DB2 Tables that are defined in the program. This field can be expanded to list the DB2 tables defined.
DB2 Tables Used	The number of DB2 Tables that are used in an SQL statement within the program logic. This field can be expanded to list the DB2 tables used.
Maintenance Effort	A measure of the effort required to maintain the program, based on program clarity, computed as the ratio of volume to level. The lower the number, the easier the program will be to maintain. This is a relative number that can be compared to other programs to determine the maintenance effort for various programs.
Unique Operators [n1]	The unique number of verbs and elements other than data elements occurring in your program. Operators are syntactic elements such as +, -, <, >.
Unique Operands [n2]	The unique number of data elements referenced in your program. Each data element is only counted the first time it is referenced. Operands consist of literal expressions, constants, and variables. This is not a count of all data elements defined, but those that are actually used in the program.
Total Operators [N1]	The total number of verbs and elements other than data elements occurring in your program. Paired operators such as BEGIN .. END, DO .. UNTIL , FOR .. NEXT are treated as a single operator.
Total Operands [N2]	The total number of data elements occurring in your program. Each time the data element is referenced it is added to this count. Operands consist of literal expressions, constants, and variables. This is not a count of all data elements defined, but those that are actually used in the program.
Vocabulary	The number of unique operators and operands in your program, n, computed n1+n2. This is an estimation of the size of the program's vocabulary (the number of things that must be known to understand the program).

Length	The length of your program, N, computed N1+N2.
Computed Length	A prediction of program length, N^, computed n1log2n1+n2log2n2. It has been experimentally observed that computed length is in close agreement to program length.
Volume	A measure of the size of a piece of code (computed Nlog2n) compared to potential volume (n*log2n*), where n* is the size of potential vocabulary. (Potential volume is independent of the language.)
Level	The ratio of potential volume to actual volume, a measure of how abstractly the program is written.
Intelligence Content	The total content of your program, computed by multiplying program level and volume. It is shows the complexity of a given algorithm independent of the language used to express the algorithm.

Related Topics

Viewing Procedure Metrics

Viewing Code Problems

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

Viewing Procedure Metrics

When you open a program in SlickEdit and the **Program Analysis** perspective, the program is parsed and analyzed to build the program structure chart and provide in-depth information about the program. The **Program Analysis** perspective includes a **Procedures** view listing metrics for all the procedures in the program and a **Properties** view listing the same metrics for just the selected procedure. Click on a different procedure node in the program structure chart to display the metrics for that procedure in the **Properties** view.

Note: The **Procedures** view, if not visible, can be opened from the **Window** menu by selecting **Show View > Other > Program Analysis > Procedures**. The **Properties** view, if not visible, can be opened from the **Window** menu by selecting **Show View > Other > General > Properties**.

The two columns of the **Properties** view, **Property** and **Value**, contain the data in the table below. The same data is consolidated for all procedures in the **Procedures** view.

Property	**Value**
Procedure Name	Name of the procedure.
McCabe	The metric that relates to the number of decision points (points where the logic path splits) in the procedure. A high number indicates that the procedure is complex. Procedure nodes in the program structure chart are highlighted in shades of red indicating their McCabe Complexity metric. The higher the number, the darker the shading, relative to the other nodes in the chart. The darkest shading indicates the most complex parts of a given program, and the lightest indicates the least complex, even though the actual numbers will vary for different programs.
Statements	The number of statements in the procedure.

Blocks	Number of code blocks in the procedure. These code blocks are found in the logic flow chart.
Conditionals	The number of conditional statements in the procedure.
GOTOs	Number of GO TO statements.
File I/O	Number of file I/O statements in the procedure.
Performs	The number of performs in the procedure.
Program Calls	The number of external program calls in the procedure.
Perform Range Violations	Indicates whether any GO TO statements branch outside their procedures, which would be a perform range violation. This column only appears for COBOL programs. PL/I programs will display the default of **No**.
Group Returns	Indicates whether control returns to the statement following the procedure. **Yes** means the procedure returns control. **No** means the procedure does not return control. **Maybe** means the procedure contains a conditionally executed GOBACK, STOP RUN, or EXIT PROGRAM and potentially does not return control.
Starting Line	Starting line of the procedure.
SQL Statements	The number of SQL calls in the procedure.

The values in the **Procedures** view can be exported for later comparison following code changes. For more information, see Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views.

Related Topics

Viewing the Program Summary

Viewing Code Problems

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

Displaying Properties

The **Properties** tab enables you to view Information about nodes and connecting lines in the program structure and data flow charts. The properties displayed depend on the item selected.

Clicking on a ***program structure chart node*** displays available values for the following on the **Properties** tab:

- Blocks
- Conditionals
- File I/O
- GOTOs
- Group Returns
- McCabe
- Perform Range Violations
- Performs
- Procedure Name
- Program Calls
- SQL Statements
- Starting Line
- Statements.

For descriptions of each of these values, see Viewing Procedure Metrics.

Clicking on a ***data flow chart node*** displays values where available for the following on the **Properties** tab:

- Field
- Defined on line
- Definition member
- Definition member dataset
- Initial value
- Length(Logical)
- Length(Physical)
- Level
- Memory offset
- Occurs
- Parent
- Picture
- Usage
- Where modified
- Where used.

Clicking on a ***program structure, logic flow, or data flow chart connector line*** displays values where available for the following on the **Properties** tab:

- Group ID
- ID
- Name
- Source and/or Destination, which are expandable to display the properties of the corresponding node(s) listed above.

Related Topics

Viewing Procedure Metrics

Working with the Program Structure Chart

Working with the Data Flow Chart

Using the Hierarchy View

Updating the Project Hierarchy

The **Hierarchy** view can aid in understanding the implications of modifying programs and copybooks in an online Compuware mainframe project. It is based on a prior analysis of the project and can be used to view either a hierarchy of copybooks or a hierarchy of program calls. You can use the copybook hierarchy to display both an *include* hierarchy and an *included by* hierarchy. The program hierarchy can display both a *calls program* hierarchy or a *called by program* hierarchy.

Project Explorer

Content for the **Hierarchy** view is created by first analyzing an online Compuware mainframe project in the **Project Explorer**. The project should include source members and a copybook concatenation list. For more information on creating and configuring an online project, see the *Topaz Workbench Host Explorer User Guide*.

Note: The project must contain either COBOL or PL/I programs, but *not* a mixture of the two.

Note: The source member names of PL/I programs must observe the convention of matching the procedure name within the member.

Note: When opening editors from either the **Project Explorer** or **Hierarchy** view, the copybooks from the project's copybook concatenation list are used, not the SlickEdit copybook concatenation lists configured in **Host Explorer**.

Update the project hierarchy for an online Compuware mainframe project

1. To start creating or updating project hierarchy data for an online project, right-click anywhere on the project and select **Update Project Hierarchy**. The entire project is analyzed to determine the hierarchical relationships of programs and copybooks. Any problems encountered during analysis are presented in the **Problems** view. This analysis can take some time, depending on the size and complexity of the project. An **Analyze project** percentage indicator in the status area shows the progress of the scan. Double-clicking the indicator displays a more detailed **Progress** view and allows you to terminate the hierarchy update process, if necessary.

2. When creation or updating of the project hierarchy is complete, right-click a program or copybook in the **Project Explorer** and select either:

   - **Open Copybook Hierarchy** or
   - **Open Program Hierarchy**

3. Continue with Working with the Hierarchy View.

Related Topics

Working with the Hierarchy View

Working with the Hierarchy View

Program Analysis presents a **Hierarchy** view tab when **Open Program Hierarchy** or **Open Copybook Hierarchy** is selected for a resource in an online Compuware mainframe project as described in Updating the Project Hierarchy. The **Hierarchy** view can show either a program hierarchy or a copybook hierarchy, depending on which was selected from **Project Explorer.** The program hierarchy can show the programs called by the selected program or the programs that call the selected program. Likewise, the copybook hierarchy can show the copybooks referenced in the selected copybook or the copybooks that reference the selected copybook. Items in the **Hierarchy** view can be expanded to show subsequent items nested below.

Programs are indicated by the program icon, and copybooks are indicated by the copybook icon. If a recursive program call exists in the project, it is indicated by the recursive program icon, and if recursive copybook inclusion exists in the project, it is indicated by the recursive copybook icon.

The table adjacent to the hierarchy provides details about the individual program calls or copybook includes in the **Hierarchy** view and can be used to open a selected resource and position to the line identified.

Viewing Program Hierarchy

1. Ensure the online mainframe project has been analyzed as described in Updating the Project Hierarchy.

2. Right-click a program in **Project Explorer** and select **Open Program Hierarchy**. The **Hierarchy** view appears, showing the program members called by the selected program. The type, name, and location of the calling program are listed at the top of the view. The calling program is shown at the top of the hierarchy, and called programs are shown indented below with the dataset listed next to each program member name. If a resource did not exist in the project when it was analyzed, **Not Found** is shown next to the name.

   Note: For PL/I procedures that are internal to a program, **Internal procedure in *program*** is shown next to the name, where ***program*** is the name of the program in which the procedure is defined.

3. Called programs can be expanded to show deeper levels within the call hierarchy. Click the arrow next to the called program. If the called program calls other programs, they are shown indented below. If there are no called programs, the arrow disappears. If there is a recursive call, the recursive program icon is displayed and the expansion arrow is not shown.

4. Click a called program to show **Line**, **Reference**, and **Calling Procedure/Paragraph** information for that program in the adjacent table.

5. Double-click any program in the **Hierarchy** view to open it and position to the first line, or double-click a table entry to open the program and position to the line of the call.

6. To switch the view to show members that call the selected program, click the **Show Called By Hierarchy** button. To switch back to the initial view, click the **Show Calls Hierarchy** button.

   Note: The most recent selection of the **Show Called By Hierarchy** button or the **Show Calls Hierarchy** button remains in effect for subsequent **Open Program Hierarchy** or **Open Copybook Hierarchy** requests for other members.

Viewing Copybook Hierarchy

1. Ensure the online mainframe project has been analyzed as described in Updating the Project Hierarchy.

2. Right-click a program or copybook in **Project Explorer** and select **Open Copybook Hierarchy**. The **Hierarchy** view appears, showing the copybook members referenced by the selected program or copybook. The type, name, and location of the referring program or copybook are listed at the top of the view. The selected program or copybook is shown at the top of the hierarchy, and referenced copybooks are shown indented below with the dataset listed next to each copybook name. If a resource did not exist in the project when it was analyzed, **Not Found** is shown next to the name.

   Note: For PL/I procedures that are internal to a program, **Internal procedure in *program*** is shown next to the name, where ***program*** is the name of the program in which the procedure is defined.

3. Referenced copybooks can be expanded to show deeper levels within the copybook hierarchy, if they exist. Click the arrow next to the referenced copybook. If the referenced copybook references other copybooks, they are shown indented below. If there are no copybooks referenced, the arrow disappears. If there is a recursive include, the recursive copybook icon is displayed and the expansion arrow is not shown.

4. Click a referenced copybook to show **Line**, **Reference**, and **Referring Procedure/Paragraph** information for that copybook in the adjacent table.

5. Double-click any copybook in the **Hierarchy** view to open it and position to the first line. If viewing copybook hierarchy for a program, double-click a table entry to open the program and position to the selected include.

6. To switch the view to show programs or copybooks that include the selected copybook, click the **Show Included by Hierarchy** button. To switch back to the initial view, click the **Show Includes Hierarchy** button.

   Note: The most recent selection of the **Show Included By Hierarchy** button or the **Show Includes Hierarchy** button remains in effect for subsequent **Open Program Hierarchy** or **Open Copybook Hierarchy** requests for other members.

Related Topics

Updating the Project Hierarchy

Exporting the Program Structure Chart, Logic Flow Chart, and Data Flow Chart

The program structure chart, logic flow chart, and data flow chart can each be exported to the following formats:

- Portable Document Format (.pdf) file for later viewing and printing.
- Visio® XML diagram (.vsdx) file for later use in Microsoft Visio®.
- Portable Network Graphic format (.png) for later viewing.
- JPEG format (.jpg) for later viewing.

To export the program structure chart, logic flow chart, or data flow chart

1. Use Program Analysis functions to tailor the content of the program structure chart, logic flow chart, or data flow chart before exporting. For more information, see Working with the Program Structure Chart, Working with the Logic Flow Chart, or Working with the Data Flow Chart.
2. Click the **Export** button on the toolbar to the upper right, or press **Ctrl+Alt+E**. The **Export** dialog box appears.
3. Select the desired export file format from the **Format** list.
4. Click **Browse** and browse to the desired location for the exported chart.
5. If **Export Image (PNG, JPEG)** was chosen from the **Format** list, select ***.png** or ***.jpg** from the **Save as type** drop-down list.
6. Enter a file name for the exported chart and click **Save**.
7. To open the saved file automatically after export, select the **View document after export** check box.
8. To optionally include the current date, select the **Include date** check box.
9. To optionally include a watermark, select the **Include watermark** check box.
10. To optionally include a title for the exported chart, type the desired title in the **Title** field. The name of the program is the default.
11. Click **Finish**. If a file with the specified name already exists, a prompt will appear to confirm overwriting the existing file. The file is generated and -- if the **View document after export** check box was selected -- opened in your system's default viewing application for the exported file type.

Related Topics

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

Exporting the Program Summary, Problems, Procedures, and Data Flow Table Views

The **Program Analysis** perspective includes a **Program Summary** view listing metrics for the overall program, a **Procedures** view listing metrics for all the procedures in the program, a **Problems** view listing any code problems identified during analysis, and a table view of the data flow chart. The contents of each of those views can exported to a delimited file. The resulting file can then be compared with a later version following code changes. This before vs. after comparison makes it easier to understand the impact of whatever changes have been made.

To export the Program Summary, Problems, Procedures, or Data Flow Table view

1. Click the **Export** button on the toolbar to the upper right, or press **Ctrl+Alt+E**. The **Export** dialog box appears.
2. Either enter a name for the exported csv file in the **File name** field, or click **Browse**, browse to the desired location, enter a name for the exported csv file in the **File name** field, and click **Save**.
3. Choose a **Field delimiter** from the dropdown list. Options are **Tab**, **Semicolon**, **Comma**, **Pipe**, and **Space**.
4. Choose a **Text qualifier** from the dropdown list. Options are **None** (the default) and **Double quote**.

5. Click **OK**.

Related Topics

Exporting the Program Structure Chart, Logic Flow Chart, and Data Flow Chart

Specifying Preferences

Data Visualizer preferences allow you to adjust the way nodes are presented in the logic flow and data flow charts. Properties for a mainframe project allow you to control the compiler/parse options related to generating the **Hierarchy** view.

Data Visualizer Preferences

To change your preferences, click **Window > Preferences**. The **Preferences** dialog box appears. Expand **Compuware** and select **Data Visualizer**. The Preferences for program analysis visualization are shown. When you change your preferences settings, the updated preferences will be remembered the next time you start Program Analysis. To restore the default settings, click **Restore Defaults**. Then click **Apply** or **OK** to accept your changes.

Limiting Node Sizes

To limit the horizontal size of nodes, select the **Limit node sizes** checkbox and specify a limit in the **Maximum displayed characters before trimming** field. Valid values are from 8 to 512 with a default value of 16. Characters beyond the specified value are indicated by an ellipsis (...) at the end of the name. The **Limit node sizes** checkbox is deselected by default. Click **Apply** or **OK** to accept your changes.

Line Wrapping Nodes

To enable names to wrap to additional lines in each node, select the **Line wrap nodes** checkbox and specify a limit in the **Minimum displayed characters before line wrap** field. Valid values are from 8 to 512 with a default value of 16. Names will wrap to a new line each time a line reaches the specified number of characters, and the node will expand vertically as needed to hold the wrapped name. The **Line wrap nodes** checkbox is selected by default. Click **Apply** or **OK** to accept your changes.

Mainframe Project Properties

To change your mainframe project properties, right-click the project and select **Properties**. The properties for COBOL and PL/I compiler/parse options are shown.

Specifying Compiler/Parse Options for a Mainframe Project

1. To specify COBOL options, first select **Enterprise COBOL** from the **Language level** dropdown list, then specify desired COBOL values for **SQL Apostrophe Quote**, and **COBOL Apostrophe Quote**. Click **Apply**.

2. To specify PL/I options, first select **PL/I** from the **Language level** dropdown list, then specify desired PL/I values for **Left Margin**, **Right Margin**, **NOT Symbol**, **OR Symbol**, and **SQL Apostrophe Quote**. Click **Apply**.

3. Click **OK** to accept your changes.

Note: Compiler/parse options are included in exported projects.

Related Topics

Working with the Logic Flow Chart

Working with the Data Flow Chart

Updating the Project Hierarchy

Runtime Visualizer

Welcome to Runtime Visualizer

Compuware's Runtime Visualizer is an Eclipse plug-in you can use to view a dynamic visualization of program and I/O calls presented in the sequence in which they occur. The events shown can be replayed, exported to various formats, and imported for replay. By observing these execution events, you can quickly achieve greater understanding of your complex, mission-critical applications while increasing your confidence in modifying or extending these applications.

The Runtime Visualizer plug-in includes the **Runtime Visualizer** perspective for working with runtime visualizations. It has an upper view that displays a graphical representation of the actual execution of your application in a debug session, and a lower section with two views, one summarizing execution events in an expandable tree and one listing individual execution events in tabular format. Views showing the call stack in graphical and text format can also be opened to provide additional insight into the sequence of events depicted in the runtime visualization.

Runtime visualization is initiated by first selecting it on the **Visualizer** tab of Xpediter/Eclipse's **Debug Configurations** dialog box, and then submitting the debug request. A completed runtime visualization can also be exported as a .csv file that can later be imported back into the Runtime Visualizer and, optionally, animated.

Runtime Visualizer works with all the environments currently supported by Xpediter/Eclipse, including batch, CICS, and IMS online. COBOL, PL/I, Assembler, and C programs are supported, subject to the limitations listed below.

Limitations

- If any non-COBOL language is involved in the program calling chain, the visualization may be incomplete. For example, not all calls from an Assembler program will be detected and shown.

- Static and dynamic program calls are supported for COBOL, but non-COBOL program calling chains can only be followed for static module calls, not dynamic ones.

- Programs called by using DLL linkage will not be detected, regardless of language.

- Although source is not required for a program to be included in runtime visualization, Xpediter/Eclipse requires standard use of DDIO libraries.

Related Topics

Getting Started with Runtime Visualizer

Getting Started with Runtime Visualizer

This topic will help you prepare to start using Runtime Visualizer.

To use Runtime Visualizer for mainframe data, you must define an HCI host connection and connect Topaz® Workbench or Eclipse/RDz to a mainframe system. To define your connection, click **Window > Preferences**. Then expand **Compuware > Host Connections** and add an HCI host connection before continuing.

A host connection between the mainframe host and Topaz Workbench should have already been defined during the installation and configuration of Topaz Workbench. If you do not have host connection information (typically a port number and a host name), contact your system administrator to obtain specific connection configurations for your site for establishing Host Connections. (For system administrators, see the *Topaz Workbench Installation Guide* for additional information on installing Topaz Workbench.)

Note: When using File-AID Services in support of File-AID Data Privacy in Topaz Workbench, Host connections are centrally configured on the server and available to all users that define File-AID Services in their Compuware preferences.

Related Topics

Initiating Debugging for Runtime Visualizer

Working with Runtime Visualizer

Initiating Debugging for Runtime Visualizer

Rather than performing a static analysis of program source files, Runtime Visualizer presents the actual execution of your application as captured during an Xpediter debugging session.

Selecting runtime visualization in Xpediter

1. Start Topaz Workbench, then open the **Xpediter** perspective as described in the *Topaz Workbench Xpediter/Eclipse User Guide*.

2. Create a new or open an existing debug configuration as described in the *Topaz Workbench Xpediter/Eclipse User Guide*.

3. On the **Visualization** tab, select the **Visualize program and I/O calls** checkbox.

4. Click **Debug** to start the debugging session. If necessary, click the **Resume** toolbar button or press **F8** to complete the debugging session.

5. When the debugging session completes, a **Runtime Visualizer** view appears showing the program and I/O calls, one after another, in the order in which they occurred.

Working with Runtime Visualizer

When the debugging session completes (see Initiating Debugging for Runtime Visualizer), a **Runtime Visualizer** view appears that can be used to show the program and I/O calls, one after another, in the order in which they occurred. Switching to the **Runtime Visualizer** perspective displays the visualization along with a lower view that presents the execution sequence in two different formats: A summary of execution events is shown in the **Runtime Visualizer Summarized Events** view, while each individual execution event is listed in the adjacent **Runtime Visualizer Individual Events** view. Two additional views, the **Call Stack** view and the **Call Stack Text** view, can be opened by right-clicking a node and selecting **Show In > Call Stack View** or **Show In > Call Stack Text View**. These views provide an easy way to look at the chain of execution events leading up to the selected event, and once opened will continuously synchronize with whatever visualization item is selected.

As the runtime visualization is being drawn, rectangular nodes representing application resources are added one after another to the diagram, connected by lines representing program and I/O call events. A root node appears first at the top of the visualization representing Xpediter's execution of the program, with the task ID (Thread ID) available on the **Properties** tab. In a CICS environment, multiple root nodes may be shown with a different CICS task ID for each.

Event connecting lines have arrowheads indicating the direction of data flow: For an input operation, such as READ or SQL FETCH, the arrowhead points from the data object to the calling object. For output operations, such as WRITE or SQL INSERT, the arrowhead points from the calling object to the data object. No arrowhead is shown if the I/O is neither clearly input or output, such an OPEN, CLOSE, or START. Event connecting lines also include a small invocation counts box showing the number of times a resource was called. You can hover over an event connecting line to display a tooltip describing the relationship represented, including the number of occurrences. The title displayed on the visualization tab is based on the first execution event's program and timestamp.

The following icons appear in the nodes of the upper runtime visualization view. Each node also includes additional information, if available:

. ⚙	**Program**. The program name and load module are shown in the node.
FILE	**File**. The file type is shown in the node.

DB2	**DB2 table**. The table name in shown in the node. Hover to display a tooltip with table name and table owner, if available.
IMS	**IMS database**. The database name is shown in the node.

Nodes in the upper runtime visualization view have different colors based on the selected metric color. Right-clicking on the chart and selecting **Metric Color** displays a menu of available node color choices with their corresponding program characteristics:

□	None
🟪	File I/O
🟩	SQLCalls
🟨	Program Calls.

When nodes are colored in shades of purple indicating their File I/O metric, the higher the number, the darker the shading relative to the other nodes in the chart. The darkest shading indicates the most file input/output occurring in a given program, and the lightest indicates the least I/O, even though the underlying numbers will vary for different programs.

Other metric colors are similarly lighter or darker based on the relative number of SQL calls or program calls present in those nodes.

Note: Selecting a single item in the **Runtime Visualizer Individual Events** view that includes calls to multiple DB2 tables will highlight event connecting lines and nodes associated with all of the calls.

Right-clicking on an object node representing an MVS file or a DB2 table displays a menu with choices for **Edit with File-AID** and **Browse with File-AID.** Choosing **Edit with File-AID** creates a File-AID Data Editor request that can be submitted to edit the selected object, and choosing **Browse with File-AID** creates a request for browsing the object. For more information, see the *Topaz Workbench File-AID/Eclipse User Guide*.

The **Runtime Visualizer** perspective also includes a **Properties** view that lists a wide variety of properties and their values for whatever event or node you select. Use the ☰ **Show Categories** button to group properties under **Event**, **Relationships**, and **Tables** categories. Only properties with available values are shown.

The **Events** views can also be added to your **Xpediter** perspective from the **Window** menu by selecting **Show View > Other**, expanding **Runtime Visualizer**, and choosing **Runtime Visualizer Individual Events** or **Runtime Visualizer Summarized Events**. The **Properties** view can be added from the **Window** menu by selecting **Show View > Other**, expanding **General**, and choosing **Properties**.

Using the Visualizer Control Buttons

While the runtime visualization is being created, you can use the ■**Stop** button to terminate the building of the visualization or the ⏸**Pause** button to temporarily halt it. A runtime visualization that is paused or has not begun building can be started with the ▶ **Play** button.

When the runtime visualization completes, the ■ **Stop** and ▶ **Play** buttons are dimmed because their actions are not available. You can return to the start of execution and step forward and backward through the entire series of execution events as follows:

- Click the ⏮ **First Event** button to return to the start of execution.
- Click the ⏭ **Last Event** button to go to the end of execution.

- Click the ◀ **Step Backward** and ▶**Step Forward** buttons to move backward and forward.

You can also use the keyboard arrow keys to move up and down through the list of events in either of the **Events** views.

Tailoring the Visualization

The completed runtime visualization can be adjusted for easier viewing. Each node can be moved to a new location by dragging it with the left mouse button pressed. The contents of the entire visualization can also be moved either by using the scroll bars or by placing the cursor on an open area then dragging with the left mouse button pressed.

Nodes can be arranged top to bottom (the default), left to right, or bottom to top by right-clicking anywhere in the runtime visualization and selecting **Layout > Top Down**, **Layout > Left to Right**, or **Layout > Bottom Up**.

Runtime visualization nodes can be collapsed to hide subsequent nodes by right-clicking and selecting **Collapse**. The collapsed node includes a red indicator in the lower right corner showing the number of levels that have been collapsed. To redisplay nodes hidden under a collapsed node, right-click the node and select **Expand**, **Expand All**, or **Expand Levels**. Selecting **Expand** expands the collapsed node one level. Selecting **Expand All** fully expands all nodes under the collapsed node. **Expand Levels** provides sub-menu choices to expand the collapsed node by 3, 5, or 7 levels. If the node has been collapsed more levels than it is expanded, subsequent nodes that remain collapsed include a red indicator showing the number of levels that could still be expanded. For easier viewing after the desired nodes have been collapsed, the visualization can be redrawn without the hidden nodes by right-clicking anywhere in the runtime visualization and selecting the desired **Layout>** option.

Viewing a Subsection of the Visualization

Large runtime visualizations can be difficult to view and navigate. To make it easier to focus on a particular section of the visualization, right-click the node you are interested in and select **View Subgraph > Make Root**. The selected node becomes the root node for a runtime visualization sub-chart, and an indicator appears in the top right corner of the node showing how many parent nodes directly above have been hidden. A node must have children to be set as root. To display one level higher, right-click the new root node and select **View Subgraph > Show Parents**. To redisplay the full runtime visualization, right-click anywhere in the chart and select **View Subgraph > Show All Nodes**. The **View Subgraph** context menu choice is also available in the **Runtime Visualizer Summarized Events** and **Individual Events** views.

Zooming and Spacing

Visualizations can be zoomed in and out individually, and the spacing between nodes can be increased and decreased, independent of the zoom level. The zoom percentage drop-down at the top of the visualization includes a number of preset zooms from 5% to 400%, along with the choices **Page** to fit the chart fully within the view, **Height** to fit the chart to the height of the view, and **Width** to fit the chart to the width of the view. Ctrl+scroll can also be used to quickly zoom in or out through the preset zoom values. A chart can also be zoomed in small increments by Alt-clicking for larger or Shift-Alt-clicking for smaller. The default is 100%.

To view information about an individual node in full size while zoomed out smaller than 100%, simply hover over the desired node. A full-size tooltip is temporarily displayed, showing the full name of the node and other information.

The numeric spinner next to the zoom level controls the spacing between nodes, with higher numbers providing more space. Available spacing values are 0 through 50. The default is 0.

Event Details

The **Runtime Visualizer Summarized Events** and **Individual Events** views include the columns listed below. Events can be sorted based on the contents of a particular column by clicking the column heading. Default sorting is by Timestamp.

- Call Type
- Timestamp
- From Type
- From Library

- From Load Module
- From Program
- From Offset
- From Compile Timestamp
- From Language
- To Type
- To Library
- To Load Module
- To Program
- To Entry Point
- To Compile Timestamp
- To Language
- Invocation Count

The **Runtime Visualizer Summarized Events** view combines multiple instances of an execution event, with each instance listing the offset of that individual call. These combined event entries can be individually collapsed and expanded.

Exporting

The upper runtime visualization view can be exported to the following formats:

- Portable Document Format (.pdf) file for later viewing and printing.
- Delimited text file (.csv)
- Visio® XML diagram (.vsdx) file for later use in Microsoft Visio®.
- Portable Network Graphic format (.png) for later viewing.
- JPEG format (.jpg) for later viewing.

For more information, see Exporting the Runtime Visualization.

The **Runtime Visualizer Summarized Events** view, **Individual Events** view, and **Call Stack Text** view can each be exported to a delimited file for later import and viewing. For more information, see Exporting the Runtime Visualizer Events Views and Call Stack Text View and Importing a Saved Runtime Visualization.

Finding Events and Entries

To find visualization items in the Summarized Events or Individual Events view

1. The **Runtime Visualizer Summarized Events** view contains an entry for every event that occurred during execution. To filter the list to show a particular resource or call, right-click the corresponding node or connecting line in the upper runtime visualization view and select **Show In > Summarized Events**. A corresponding filter is created in the text box at the top of the **Runtime Visualizer Summarized Events** view, and the view lists only those resources or calls that match the filter. In the same way, a resource or call can be located in the **Runtime Visualizer Individual Events** view by right-clicking a node or connecting line and selecting **Show In > Individual Events**.

2. To manually specify a resource or event, begin typing its name in the text box at the top of either the **Runtime Visualizer Summarized Events** or **Individual Events** view. As you type, the list of events is filtered to list only those resources or calls that match the entered text.

3. To clear the text box in either view, click the **Clear** button to the right.

To find a Summarized or Individual Events view item in the visualization

1. The upper runtime visualization view can be large and complex. To find a particular event in the view, simply click the corresponding entry in either the **Runtime Visualizer Summarized Events** or **Individual Events** view. The nodes and connecting line involved in the event are highlighted in blue.

2. To find a particular from call, to call, or event in the upper runtime visualization, right-click the corresponding entry in the **Runtime Visualizer Summarized Events** view, expand **Find**, and select **Find Event**, **Find From Call**, or **Find To Call**.

To show an Event in another Event view or Call Stack view

1. To show a particular event from the **Summarized Events** view in the **Individual Events** view, right-click the event and select **Show In > Individual Events**.

2. To show a particular event from the **Individual Events** view in the **Summarized Events** view, right-click the event and select **Show In > Summarized Events**.

3. To show a particular event from either of the **Events** views in the **Call Stack** or **Call Stack Text** view, right-click the event and select either **Show In > Call Stack View** or **Show In > Call Stack Text View**.

Importing a Saved Runtime Visualization

After a runtime visualization that has been exported to a delimited file from the **Runtime Visualizer Summarized Events** view, that file can be imported back into the **Runtime Visualizer** for later viewing and interaction.

To import a saved runtime visualization

1. From the **File** menu, select **Import**.

2. On the **Import** dialog box, expand **Compuware** and select **Runtime Visualization**.

3. Click **Next**.

4. Click **Browse**, browse to and select the desired .csv file previously exported from the **Runtime Visualizer**, then click **Open**.

5. To view the runtime visualization as an animation, select the **Animate every *nnnn* events** checkbox. Optionally change the default of **1000** for number of events to animate. The smaller the number of events specified, the more detailed and lengthy the animation will be.

6. Specify whether to initially pause or play the imported visualization. To automatically begin building the runtime visualization on import, select **Play**. To wait until the ▶**Play** button is selected before beginning to build the visualization, select **Paused**.

7. Click **Finish**. The saved runtime visualization is imported into the **Runtime Visualizer** perspective and, optionally, animated during presentation..

Related Topics

Exporting the Runtime Visualization

Exporting the Runtime Visualizer Events Views

Exporting the Runtime Visualization

The graphical runtime visualization can be exported to the following formats:

- Portable Document Format (.pdf) file for later viewing and printing.
- Delimited text file (.csv)
- Visio® XML diagram (.vsdx) file for later use in Microsoft Visio®.
- Portable Network Graphic format (.png) for later viewing.
- JPEG format (.jpg) for later viewing.

To export the runtime visualization

1. Use Runtime Visualizer functions to tailor the content of the runtime visualization before exporting. For more information, see Working with Runtime Visualizer.
2. Right-click and select **Export**, or press **Ctrl+Alt+E**. The **Export Visualization** dialog box appears.
3. Select the desired export file format from the **Format** list.
4. Click **Browse** and browse to the desired location for the exported visualization.
5. If **Export Image (PNG, JPEG)** was chosen from the **Format** list, select ***.png** or ***.jpg** from the **Save as type** drop-down list.
6. Enter a file name for the exported visualization and click **Save**.
7. If exporting to a delimited text file (.csv), go to step 12.
8. To optionally include a title for the exported visualization, type the desired title in the **Title** field.
9. To optionally include the current date, select the **Include date** check box.
10. To optionally include a watermark, select the **Include watermark** check box.
11. To open the saved file automatically after export, select the **View document after export** check box.
12. Click **Finish**. If a file with the specified name already exists, a prompt will appear to confirm overwriting the existing file. The file is generated and -- if the **View document after export** check box was selected -- opened in your system's default viewing application for the exported file type.

Related Topics

Exporting the Runtime Visualizer Events Views

Exporting the Runtime Visualizer Events Views and Call Stack Text View

The contents of the **Runtime Visualizer Summarized Events** view, the **Individual Events** view, and the **Call Stack Text View** can each be exported to a delimited file. The resulting file can then be compared with a later version following code changes. This before vs. after comparison makes it easier to understand the impact of whatever changes have been made.

To export a Runtime Visualizer Events or Call Stack Text view

1. Click the **Export** button on the toolbar to the upper right, or press **Ctrl+Alt+E**. The **Export** dialog box appears.
2. Either enter a name for the exported csv file in the **File name** field, or click **Browse**, browse to the desired location, enter a name for the exported csv file in the **File name** field, and click **Save**.
3. Choose **Comma** from the **Field delimiter** dropdown list.
4. Choose a **Text qualifier** from the dropdown list. Options are **None** (the default) and **Double quote**.
5. Click **OK**. The exported file can be imported back into the **Runtime Visualizer** for later viewing and interaction.

Related Topics

Importing a Saved Runtime Visualization

Exporting the Runtime Visualization

Index