Tutorial

COBOL Analyzer 5.0

Advanced scenarios with COBOL Analyzer

MICRO FOCUS

Contents

Module 7: Working with Analysis Tools

Introduction

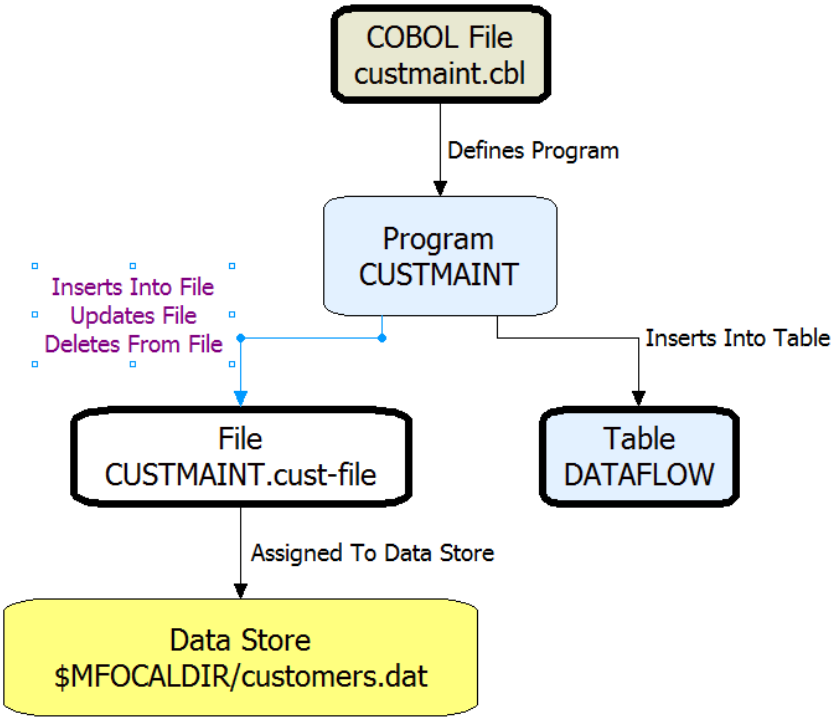
This module will get you started using the main analysis tools COBOL Analyzer provides.

Objectives

- You will learn how to use the Diagrammer tool
- You will learn how to define relationships you want to diagram
- You will learn how to use the Interactive Analysis (Hyperview) tool
- You will learn how to use code searches and code search reports
- You will learn how to use the Glossary to review conventions
- You will learn how to add Business names to data items
- You will learn how to use the Change Analyzer

The Diagrammer Tool

The Diagrammer tool helps identify and document entity relationships and the flow of data between them. The Diagrammer tool maps out the way objects interact and impact each other by presenting that information in a visual format.



For example:

In the diagram shown above, the COBOL file (custmaint.cbl) defines the Program (CUSTMAINT); the Program INSERTS, UPDATES and DELETES the DataPort (CUSTMAINT.cust-file), which is assigned to the DataStore ($MFOCALDIR/customers.dat); The program also INSERTS into the DATAFLOW table.

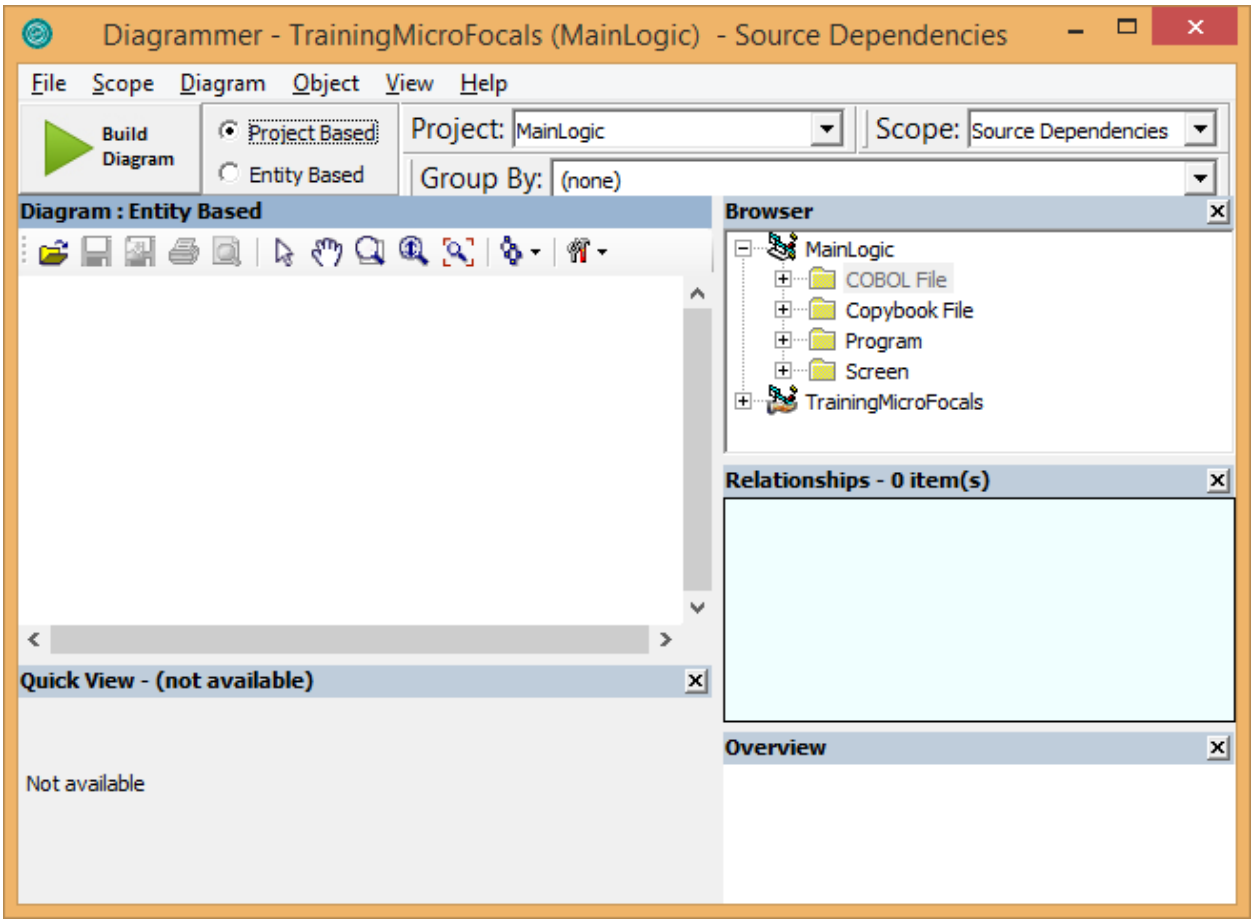
Diagrammer uses scopes (relationship definitions) to determine what types of diagrams it will generate and display. CA comes with default scopes, some of which are:

- Call Map — program to program calls
- Screen Flow — program to screen send / receive
- Source dependencies – Relationships between source files

Exercise 7-2: Viewing Object Relationships in a Project

Objective: You will become familiar with the various tools in the Diagrammer

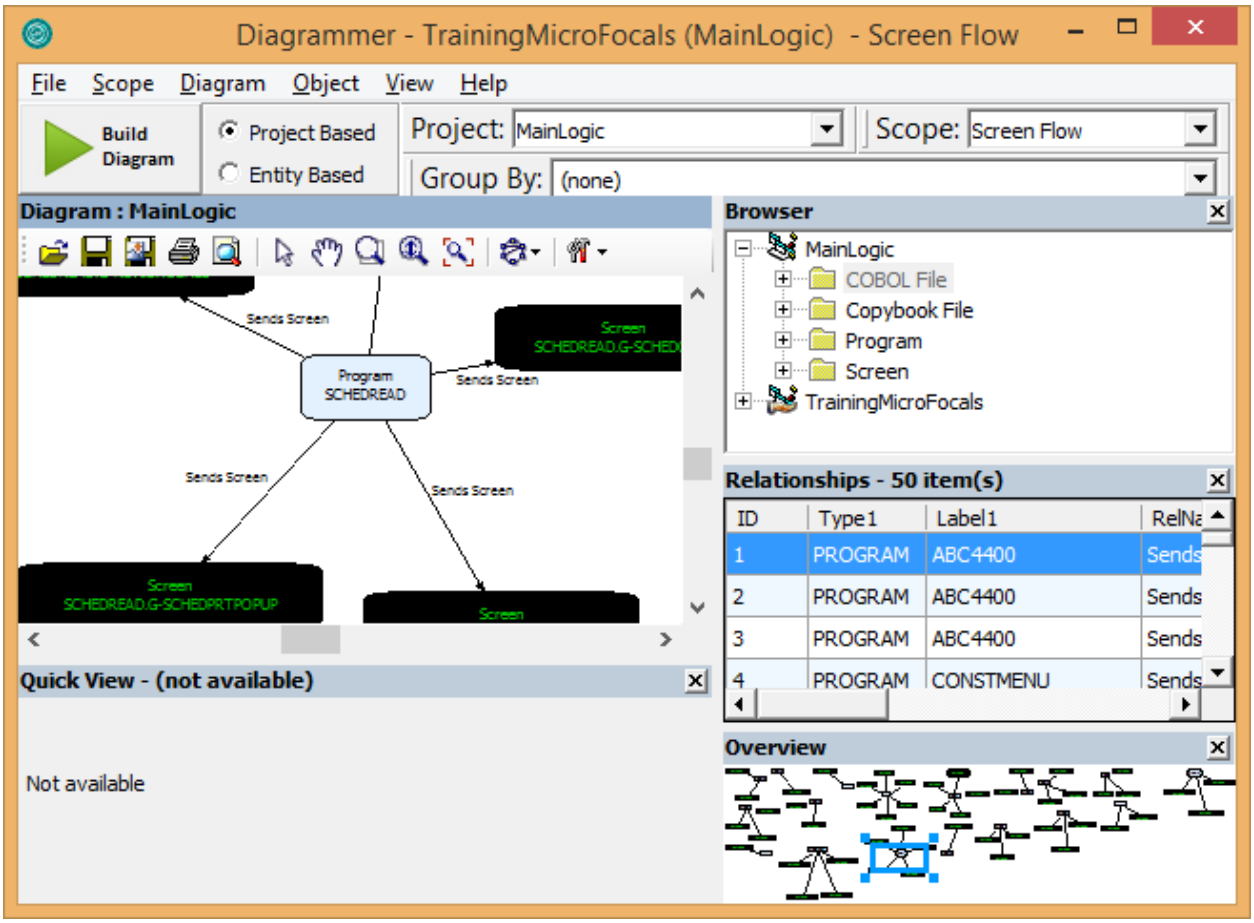
1. On the menu bar, click **Analyze > Diagrammer.** You will see the Diagrammer window. When it first opens, it will be empty.



Panes in the Diagrammer screen are:

- **Diagram** -- area to contain the diagram
- **Browser** -- project / folder structure to navigate and select Project. Also, used to select entity(s) for Entity based diagrams.
- **Relationships** – report representation of object relationships in the diagram
- **Quick View** – interactive analysis pane for selected object in the diagram (if available). Contents depend on type of entity selected.
- **Overview** -- used to 'pan' through diagram based on contents within box

2. Click the drop-down arrow next to **Scope** and select **Screen Flow**.
3. Click the  button. This will generate the diagram for the entire project based on your scope selection.



You can use the default, Project Based, or manually select one or more entities for Entity Based diagrams.

This scope generates a diagram that dis-plays program-to-screen calls within a system. The diagrams can be saved or exported in several formats including: Visio (if installed on the machine); JPEG, GIF, EMF, BMP, etc.

The Relationships pane displays the information in a table format. This List can be saved and exported to a deliverable format, much like any other report in CA.

The Overview pane displays a compact view of the Diagram. This is very useful in cases of viewing and navigating large, complex diagrams. We will see this when we use the Zoom with Marquee feature to “Pan” the diagram by grabbing the blue box in the Overview pane and positioning it over an object of interest.

Diagrammer Tools

There are many tools used in the Diagrammer. Their descriptions are below.

Navigation and Sizing Tools

The selection and zoom buttons on the Diagram pane tool bar can be used to navigate and zoom in/out on various parts of the Diagram.

Tool/Button	Description
	The **Select** tool allows you to choose specific entities on the Diagram and perform actions on them.
	The **Pan** tool can be used to scroll the diagram.
	The **Zoom with Marquee** tool can be used to select a certain area of the Diagram using click-and-drag. Diagrammer will zoom in the selected area.
	The **Zoom Interactively** tool uses click-and-drag to zoom in and out of the entire diagram as opposed to a certain area.
	The **Fit In Window** tool will reset the zoom to a size that will allow the entire Diagram to be viewed in the window.

Layout Tools

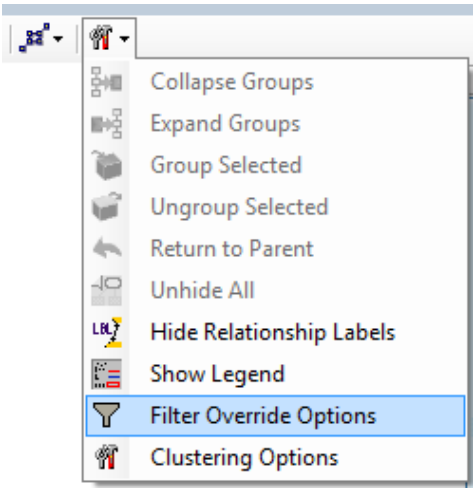
The layout buttons on the Diagram pane tool bar drop down list can be used to display the Diagrams in various styles.

Tool/Button	Description
	The **Circular Layout** tool organizes nodes in clusters of related objects.
	The **Hierarchical Layout** tool organizes nodes in precedence relationships, with levels for parent and child object types.
	The **Orthogonal Layout** tool organizes nodes in relationships drawn with horizontal and vertical lines only.
	The **Symmetric Layout** tool (the default style) organizes nodes based on symmetries detected in their relationships.
	The **Tree Layout** tool organizes nodes in parent-child relationships, with child nodes arranged on levels farther from the root node than their parents.

NOTE: The Symmetrical layout is most efficient in drawing large diagrams.

Click the Tool drop-down to select other options for your diagram. The greyed-out options are only available for objects that are manually "Grouped" by selecting individual objects and/or drawing a box around multiple objects in the diagram and choosing Group Selected. Please see online "Help" for descriptions of "Group" options.

Show / Hide Relationship Labels toggles between displaying and not displaying the Relationship Name on the connecting lines between objects in the diagram. *Show Legend* displays the meaning of the line colors and shapes in the diagram.



Filter Override Options displays the filter options that can be substituted for the default values. You can override the threshold for each. Options are:

- Hide Utilities to hide objects with many inputs, DATEFMT, for example.
- Hide Drivers to hide objects with many outputs such as a startup program.
- Hide High Input Nodes to hide objects with many inputs that do not meet the incoming relationship threshold for utilities.
- Hide High Output Nodes to hide objects with many outputs that do not meet the outgoing relationships threshold for drivers.

Defining Relationships

As mentioned before, scopes are relationship definitions used by Diagrammer to deter-mine what objects it displays. CA comes with default scopes, but also gives you the ability to create and modify your own scopes. You may want to exclude data stores from a data flow diagram or include COBOL source files in a Call Map diagram, for example. In either case, you specify the relationships that you want shown to match your desired focus of analysis. Relationships are of two types, or classes:

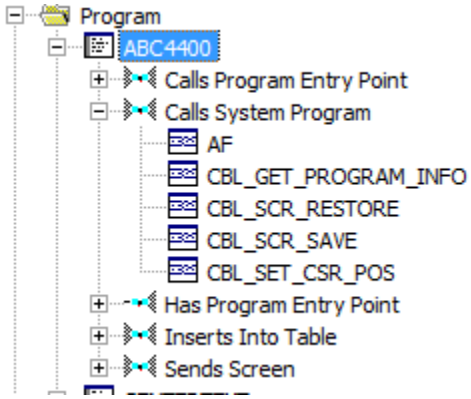
- A *basic* relationship defines the direct interaction of two objects — a program and a program entry point, for example.
- A *composite* relationship defines the indirect interaction of two or more objects — if a COBOL program inserts into a file that is assigned to a data store, for example, the relationship between the COBOL program and data store is said to be composite.

For convenience, you can create custom composite relationships that define an entire chain of relationships for an object.

Exercise 7-3: Defining Relationships to Diagram

Objective: You will define several relationships among objects and draw them in the Diagrammer

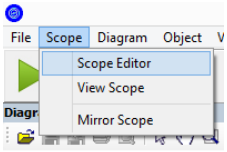
1. In the Browser pane, click on the **Program** folder and expand the **ABC**4400 entry by clicking on the + signs until you can see the relationships displayed here. Your screen should look similar to the following:



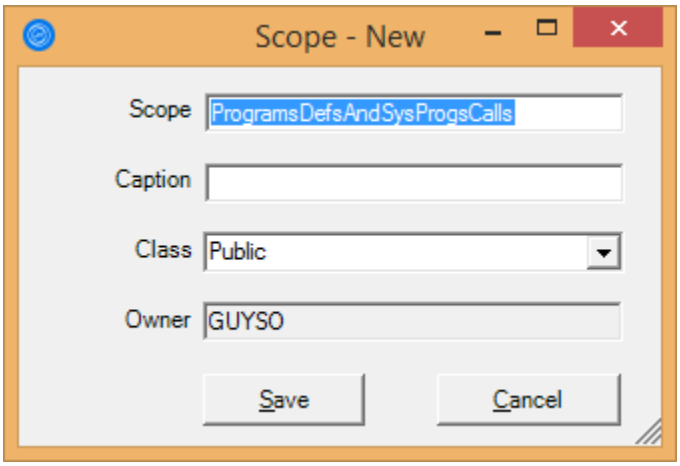
   Using the Repository pane, you see that Program ABC4400 calls various system programs.

   The goal is to display these relationships, and Program definitions in COBOL source files, visually using Diagrammer.

2. From the Diagrammer tool, click **Scope > Scope Editor** to display the **Edit Scope** window.



3. To manually create a scope (since there is not a built-in scope that contains the relationship between COBOL programs and System programs usage)
   a. From the menu bar, click **File > New Scope**. You will see the **Scope – New** dialog box.

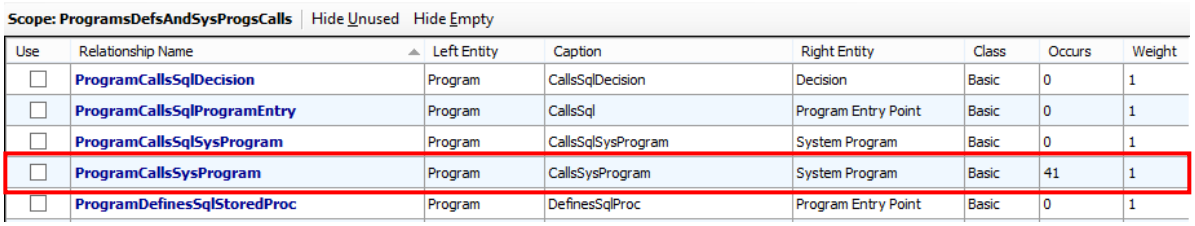


b. In the **Scope** field, type ***ProgramsDefsAndSysProgsCalls***. The **Caption** defaults to the Scope Name if not provided, but this is the value that appears in the Scope drop-down list box. Class specifies whether the scope is shared or private. The default is Public.

NOTE: Scope names must be unique in the workspace, regardless of whether they are Public or Private.

c. Press the **Save** button.

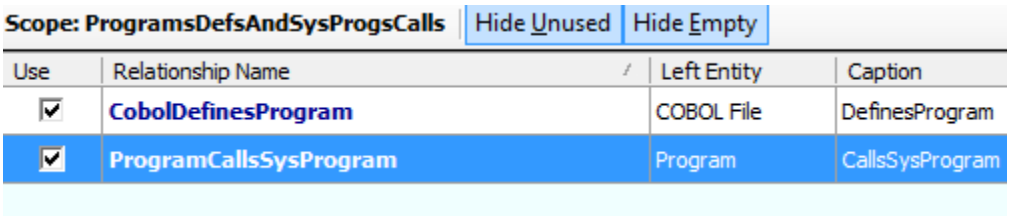
4. Click on the **Relationship Name** column header to sort the column and locate ProgramCallsSysProgram.



Use	Relationship Name	Left Entity	Caption	Right Entity	Class	Occurs	Weight
☐	ProgramCallsSqlDecision	Program	CallsSqlDecision	Decision	Basic	0	1
☐	ProgramCallsSqlProgramEntry	Program	CallsSql	Program Entry Point	Basic	0	1
☐	ProgramCallsSqlSysProgram	Program	CallsSqlSysProgram	System Program	Basic	0	1
☐	ProgramCallsSysProgram	Program	CallsSysProgram	System Program	Basic	41	1
☐	ProgramDefinesSqlStoredProc	Program	DefinesSqlProc	Program Entry Point	Basic	0	1

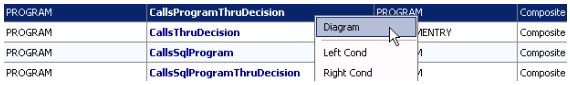
5. Check the box next to the entry.
6. Click the **Relationship Name** column header once again and sort in descending sequence. You can click the **Hide Unused** button to see only those relationships that you have selected.
   You can also click the **Hide Empty** button to see only those that are found in the workspace.
7. Scroll down to the Program section and check the COBOL defines programs.

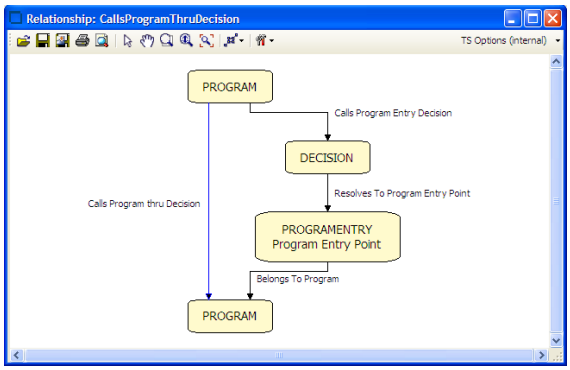
If you click the **Hide Unused** button, your screen should look similar to the following:



Use	Relationship Name	Left Entity	Caption
☑	CobolDefinesProgram	COBOL File	DefinesProgram
☑	ProgramCallsSysProgram	Program	CallsSysProgram

8. Press the **Hide Unused** button again to redisplay all relationships.
9. Right-click the **CallsProgramThruDecision** composite relationship and select **Diagram**. Do NOT check the box to include the relationship. You will see a screen similar to the following:





You will see that although the relationship displays only the endpoints (PROGRAMS), there are 2 relationships in between. This is the "composite" relationship that we are going to demonstrate by using the Basic relationships.

A composite relationship defines the indirect interaction of two objects.

To create a composite relationship, you would choose New Relationship in the File menu. The New Relationship Wizard opens. Enter the name of the composite relationship. In the Select first entity pane, select the object you want to appear on the left side of the relationship chain. Click Next. The New Relationship – Program Composite screen appears.

In the Select relationship pane, select the relationship you want to appear next in the relationship chain. Click Next. The New Relationship – Program Composite screen appears. Repeat this step for each relationship you want to include in the chain.

To delete a composite relationship, select it and choose Delete relationship in the Relationship menu. You are prompted to confirm the deletion. Click Yes.

Note: You cannot create or delete a basic relationship.

10. Close the View Relationship window by clicking the **X** in the upper right-hand corner.
11. Click **File > Save**.
12. Click **File > Exit**.
13. Click the drop-down box next to **Scope** and click "Your" ***ProgramsDefsAndSysProgsCalls***. Make sure that the Project Based radio button is selected.

14. Click the  button to build the diagram for the current project. This will generate the Diagram for the entire Project based on your Scope selection.
15. Change layouts using the Layout button  in the diagram pane toolbar and take a look at the resulting diagrams.

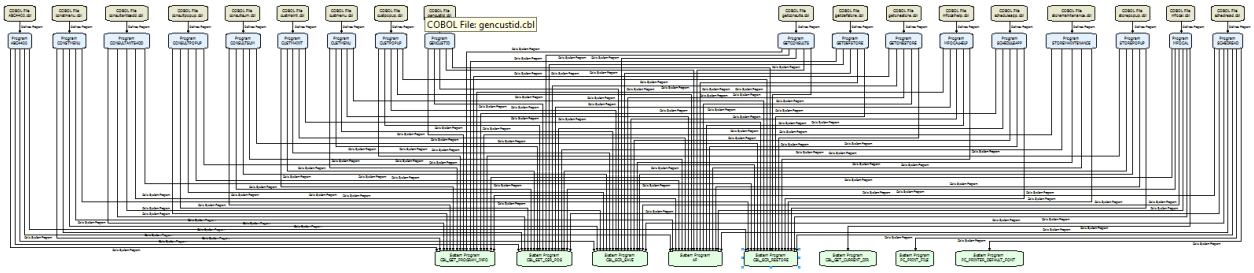


Diagram drawn with Hierarchical layout

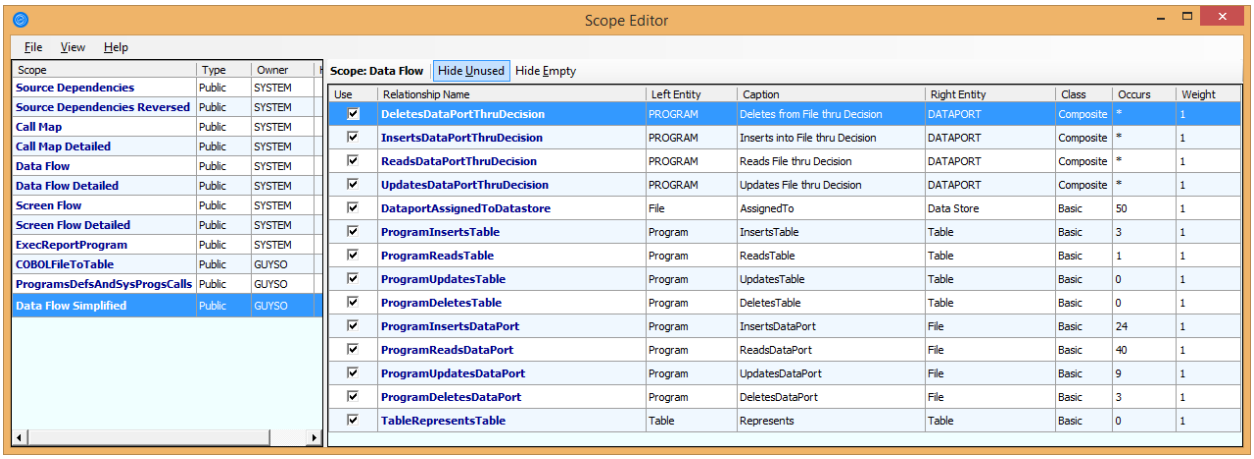
Observe the diagram and list that was produced. Notice that the diagram is a visual representation of the relationships we saw earlier in the Repository.

We can see all the system programs usages by the different programs and which COBOL files define them.

16. Click on the Zoom Interactively button and place your cursor on an object.
17. Left-click, and while holding the button, move the cursor down the screen to increase the magnification.
18. You can also use the standard Ctrl + Mouse wheel to zoom in and out
19. Double-click on the Diagram pane title and it will maximize, double-click again to restore the pane. This will work for all the Diagrammer panes (and Interactive analysis ones as well)
20. Click on the one of the programs or COBOL files entity.

Clicking on an entity populates the Quick View pane. There are several views available in the Quick View pane that will display relevant information about the selected entity.

21. Change the scope to Data Flow and Build Diagram ()
22. The diagram is too busy, let's simplify:
23. Open Scope→Scope Editor
24. Select the Data Flow scope
25. Right-click and copy
26. Change the scope name to Data Flow Simplified and save
27. Choose Data Flow Simplified scope and click the **Hide Unused** button:



Scope	Type	Owner
Source Dependencies	Public	SYSTEM
Source Dependencies Reversed	Public	SYSTEM
Call Map	Public	SYSTEM
Call Map Detailed	Public	SYSTEM
Data Flow	Public	SYSTEM
Data Flow Detailed	Public	SYSTEM
Screen Flow	Public	SYSTEM
Screen Flow Detailed	Public	SYSTEM
ExecReportProgram	Public	SYSTEM
COBOLFileToTable	Public	GUYSO
ProgramsDefsAndSysProgsCalls	Public	GUYSO
Data Flow Simplified	Public	GUYSO

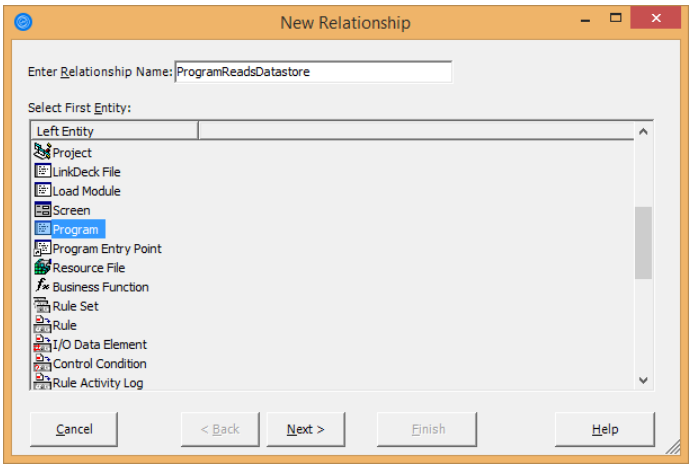
Use	Relationship Name	Left Entity	Caption	Right Entity	Class	Occurs	Weight
☑	DeletesDataPortThruDecision	PROGRAM	Deletes from File thru Decision	DATAPORT	Composite	*	1
☑	InsertsDataPortThruDecision	PROGRAM	Inserts into File thru Decision	DATAPORT	Composite	*	1
☑	ReadsDataPortThruDecision	PROGRAM	Reads File thru Decision	DATAPORT	Composite	*	1
☑	UpdatesDataPortThruDecision	PROGRAM	Updates File thru Decision	DATAPORT	Composite	*	1
☑	DataportAssignedToDatastore	File	AssignedTo	Data Store	Basic	50	1
☑	ProgramInsertsTable	Program	InsertsTable	Table	Basic	3	1
☑	ProgramReadsTable	Program	ReadsTable	Table	Basic	1	1
☑	ProgramUpdatesTable	Program	UpdatesTable	Table	Basic	0	1
☑	ProgramDeletesTable	Program	DeletesTable	Table	Basic	0	1
☑	ProgramInsertsDataPort	Program	InsertsDataPort	File	Basic	24	1
☑	ProgramReadsDataPort	Program	ReadsDataPort	File	Basic	40	1
☑	ProgramUpdatesDataPort	Program	UpdatesDataPort	File	Basic	9	1
☑	ProgramDeletesDataPort	Program	DeletesDataPort	File	Basic	3	1
☑	TableRepresentsTable	Table	Represents	Table	Basic	0	1

28. Uncheck **DataportAssignedToDataStore** (notice that it has the highest number of occurrences)
29. File→Save
30. File→Close
31. Build Diagram again

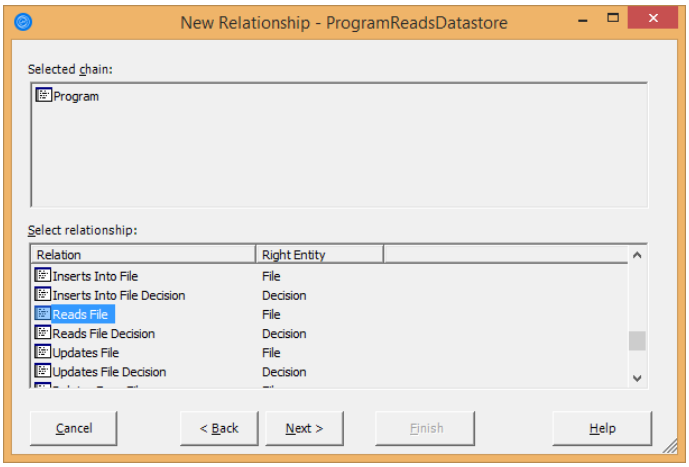
The diagram is not as busy now, but lacks the relationships between programs, we'll fix that in the next exercise.

Exercise 7-4 Creating new relationships

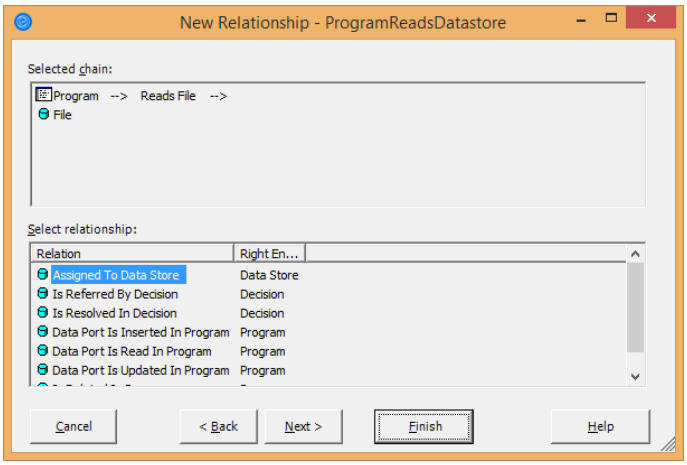
1. Open the scope editor (Scope→Scope Editor)
2. File→New relationship
3. In the relationship name enter: ProgramReadsDatastore
4. Select Program as first entity and click Next



5. Choose relation: Reads file and click next



6. Choose relation: Assigned to Data Store and click next



7. Click finish
8. Repeat steps 3-7 to create relationships for Insert, Update and Delete.
   In step 5 you'll need to select Inserts Into File, Updates File and Deletes From File.
9. Choose the Data Flow Simplified scope we created before and replace the Read, Update, Insert and Delete to data port relationships with the ones you created.
   It should look like this:

Use	Relationship Name	Left Entity	Caption	Right Entity	Class
☑	DeletesDataPortThruDecision	PROGRAM	Deletes from File thru Decision	DATAPORT	Composite
☑	InsertsDataPortThruDecision	PROGRAM	Inserts into File thru Decision	DATAPORT	Composite
☑	ReadsDataPortThruDecision	PROGRAM	Reads File thru Decision	DATAPORT	Composite
☑	UpdatesDataPortThruDecision	PROGRAM	Updates File thru Decision	DATAPORT	Composite
☑	ProgramInsertsTable	Program	InsertsTable	Table	Basic
☑	ProgramReadsTable	Program	ReadsTable	Table	Basic
☑	ProgramUpdatesTable	Program	UpdatesTable	Table	Basic
☑	ProgramDeletesTable	Program	DeletesTable	Table	Basic
☑	TableRepresentsTable	Table	Represents	Table	Basic
☑	ProgramInsertsToDatastore	PROGRAM	ProgramInsertsToDatastore	DATASTORE	Composite
☑	ProgramUpdatesDatastore	PROGRAM	ProgramUpdatesDatastore	DATASTORE	Composite
☑	ProgramDeletesDatastore	PROGRAM	ProgramDeletesDatastore	DATASTORE	Composite
☑	ProgramReadsDatastore	PROGRAM	ProgramReadsDatastore	DATASTORE	Composite

10. File→Save
11. File→Close
12. Build Diagram again

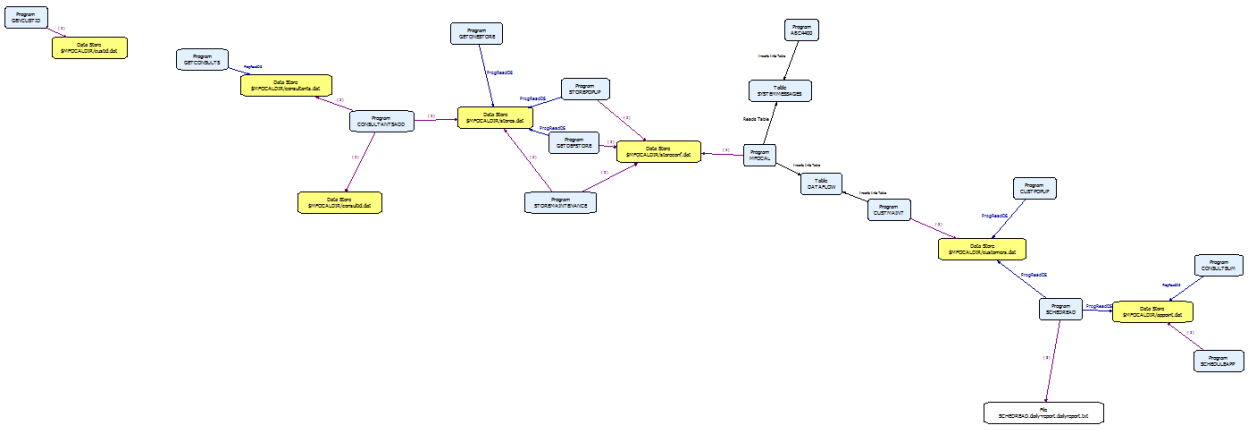


Diagram drawn with Symmetric layout

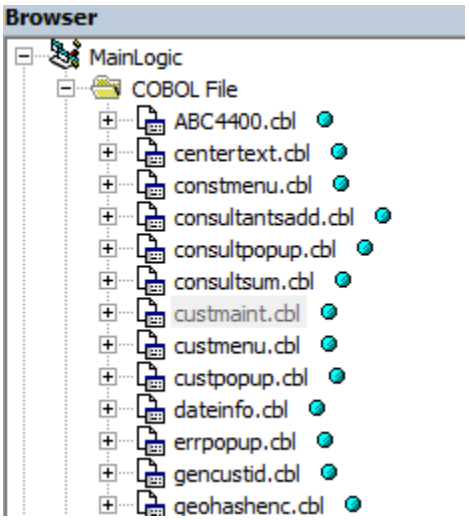
Now you can easily see which files are used by which programs.

Exercise 7-4: Viewing Entity-Based Diagrams

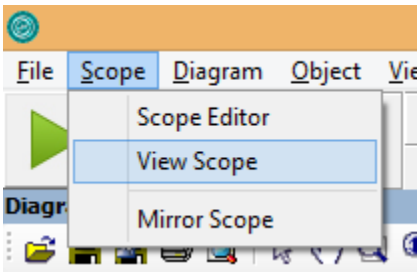
Diagrammer has a feature that lets you view relationships for selected objects only, rather than over an entire project or workspace. This enables you to view the same scopes on a much smaller and more customized scale.

Objective: You will generate an entity-based diagram

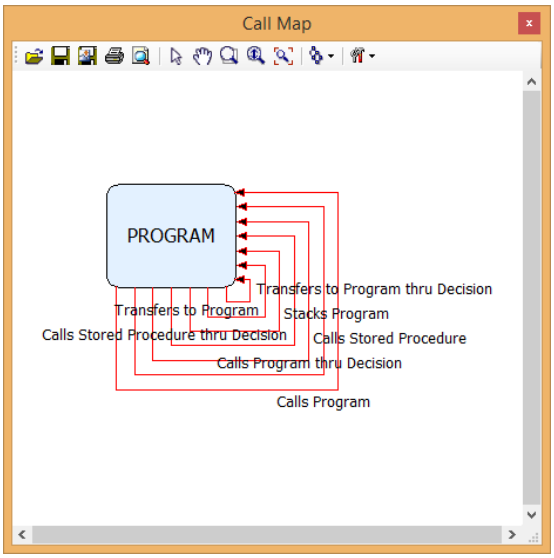
1. Expand the **COBOL File** folder in the browser window by clicking the **+** sign next to it.



2. Click the drop-down box next to **Scope** and click on **Call Map**.
3. Select **Scope > View Scope**.



This will display a template for the chosen scope. Another way to see what a scope will show it to select **Scope > Edit Scope** and sort on relationships to see which ones are checked.



The Call Map scope shows which programs are calling which programs using various calling methods. Using the same scaling controls in the pop-up window, you can alter the size and layout of the diagram.

4. Close the **View Scope** window by clicking the **X** in the top right corner.
5. Copy and paste the custmaint.cbl entity into the Diagram area by setting the focus in the browser pane and pressing **CTRL+C**. Set focus in the diagram window by clicking in the diagram window and pressing **CTRL+V**. Switch to hierarchical layout and you will see a screen similar to the following.

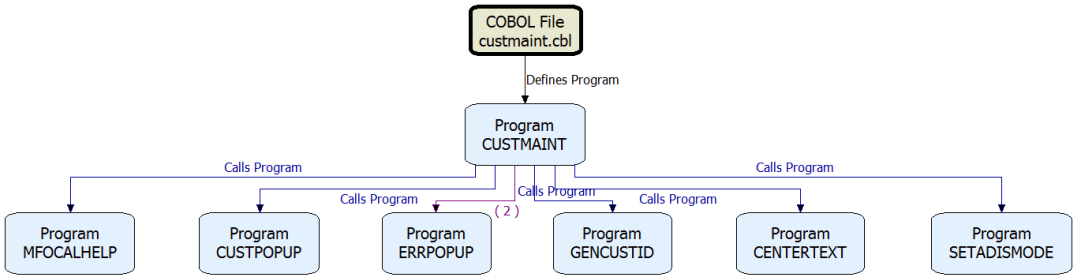
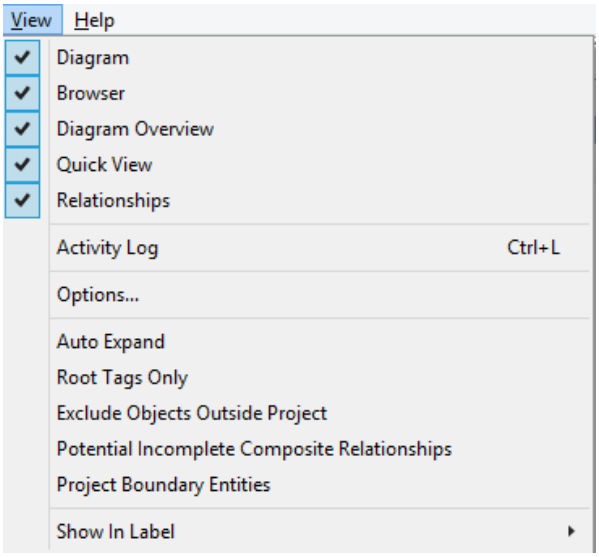


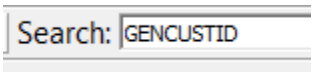
Diagram drawn with Hierarchical layout

This Call Map diagram is only for the selected program, as opposed to what you would see in Project-Based Diagram, which shows all the programs and their flow. This allows for more precise and specific diagrams on a much smaller scale.

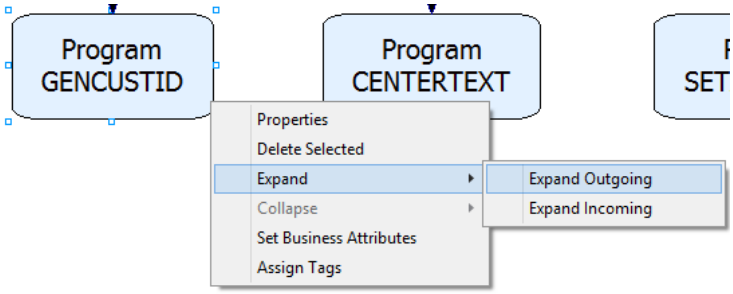
By default, the Entity Diagram is designed to only navigate one level in the relationship chain. From here, you can manually build upon the existing Diagram. This is the default View setting for Diagrammer.



6. To quickly find program **GENCUSTID**, type the program name in the search box in the upper-right corner of the window and press enter. This will place the cursor on the selected object.



7. Right-click on the **GENCUSTID** Program entity and choose **Expand Outgoing**.



Notice how we are manually traversing down the relationship chains, as expected in the definition in the View Scope window.

8. Right-click on the **CUSTPOPUP** Program entity and choose **Expand Outgoing**.

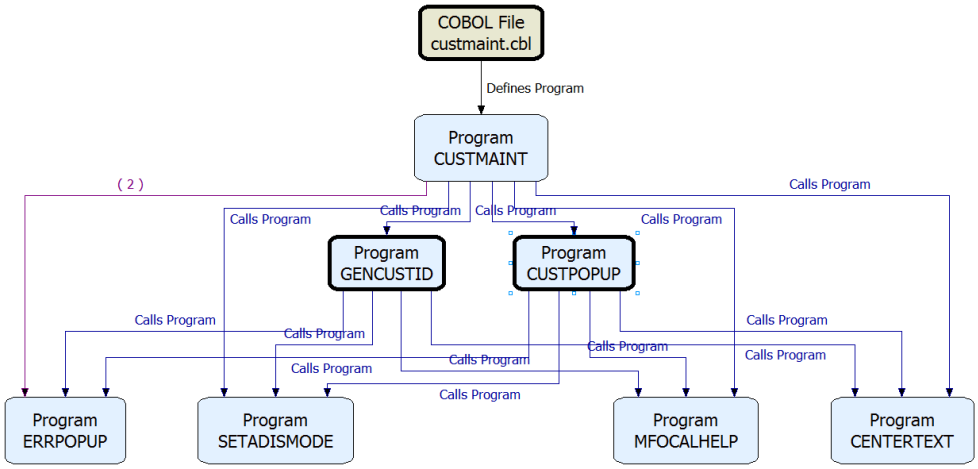


Diagram drawn with Hierarchical layout

9. Right-click on the **GENCUSTID** Program entity and choose **Collapse Outgoing.**

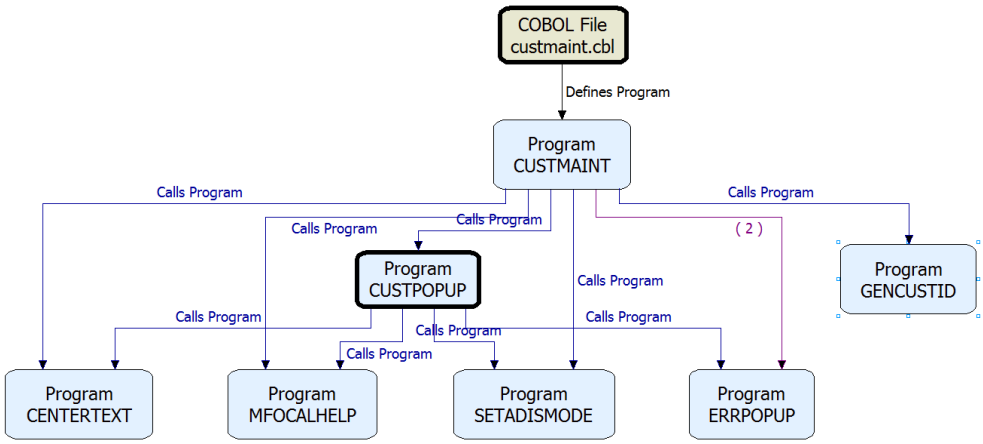


Diagram drawn with Hierarchical layout

10. Right-click on the **CUSTMAINT** Program entity and choose **Expand Incoming**.
11. Right-click on the **CUSTMENU** Program entity and choose **Expand Incoming**.

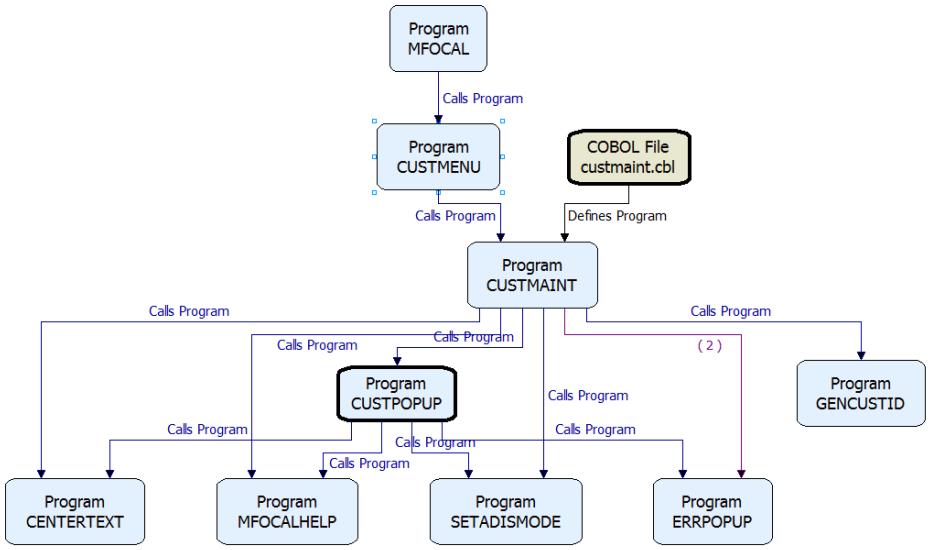


Diagram drawn with Hierarchical layout

We now see a complete call flow from the startup program (MFOCAL) to CUSTPOPUP

Relationships - 14 item(s)

ID	Type1	Label1	RelName1	Type2	Label2
1	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	CENTERTEXT
2	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	FERRPOPUP
3	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	MFOCALHELP
4	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	SETADISMODE
5	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	ERRPOPUP
6	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	CUSTPOPUP
7	PROGRAM	CUSTMAINT	Calls	PROGRAMENTRY	GENCUSTID
8	PROGRAM	CUSTPOPUP	Calls	PROGRAMENTRY	CENTERTEXT
9	PROGRAM	CUSTPOPUP	Calls	PROGRAMENTRY	FERRPOPUP
10	PROGRAM	CUSTPOPUP	Calls	PROGRAMENTRY	MFOCALHELP
11	PROGRAM	CUSTPOPUP	Calls	PROGRAMENTRY	SETADISMODE
12	PROGRAM	CUSTMENU	Calls	PROGRAMENTRY	CUSTMAINT
13	PROGRAM	MFOCAL	Calls	PROGRAMENTRY	CUSTMENU
14	COBOL	custmaint...	DefinesProg...	PROGRAM	CUSTMAINT

12. Click on the other diagram layouts by clicking on their icons.
13. When you are finished, select **Diagram > Clear** to clear the diagram window.

Interactive Analysis (HyperView)

Much of the power of CA resides in a set of program analysis tools collectively called Interactive Analysis or HyperView. HyperView lets you analyze legacy programs interactively by examining synchronized, complementary views of the same information — source, context, impacts, etc. Use HyperView to analyze procedure and data flows, stage program analyses, and extract business rules (if BRM is installed). HyperView pro-vides these panes:

- The **Objects** pane provides a list of the source objects in the project. The list can be used to navigate to other objects and object types.
- The **Source** pane displays view-only source code for the selected file and included files. Sophisticated search facilities let you navigate to code constructs quickly.
- The **Context** pane displays the parse tree for the selected source file. The parse tree displays source code constructs — sections, paragraphs, statements, conditions, variables and so forth — in hierarchical form, making it easy to locate code constructs quickly. Constructs are displayed in the order they appear in your code.
- The **Code Search** pane lets you create lists of candidates for business rule extraction, event injection, impact analysis, and other tasks. Each list captures the results of a different stage of your analysis and serves as input for subsequent tasks.
- The **Bird's Eye** pane works with the Source pane to let you quickly identify the location of a code construct relative to the entire program.

- The **Animator** allows you to step through the code displayed in a HyperView pane. You can choose program branches or have the animator choose them randomly.
- The **Program Control Flow** pane displays a diagram that shows process flow and call relationships in a program.
- The **Flowchart** pane displays a diagram of the flow of control between statements in a paragraph.
- The **Model** pane displays the parse tree meta-model in text and diagram form.
- The **Glossary** pane creates a conveniently organized dictionary, or glossary, of data elements within your application. It allows the user to associate business names and descriptions to those data elements.
- The **Watch Pane** displays program model context information in a summarized view. This is an advanced topic and will not be covered in this manual.
  The **Impact** pane window displays an impact trace for a program variable. An impact trace describes the flow of data to or from a variable. This tool is covered in the Advance features section of this training manual.
- The **Execution Path** pane displays a hierarchical view and diagram of the conditions and values that must be true in order to reach a line of code in a program.
- The **Data View** pane displays program variable structures, substructures, and fields in a hierarchical grid that visually represents memory usage.
- The **Data Flow** pane displays a diagram of the incoming and outgoing data flows for a program variable up to a data port — an I/O statement or a call to or from another program. It also displays a list of variable offsets and memory allocations.

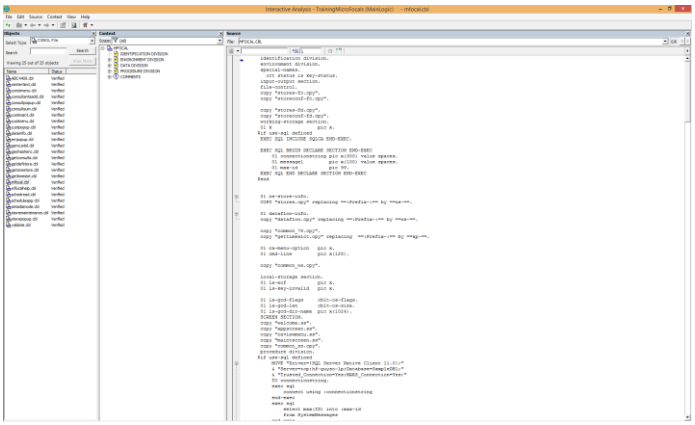
NOTE: To use HyperView, make sure that Enable HyperView is checked under COBOL File in the Verification > Settings tab in Workspace Options (it is checked by default). This is an administrative function.

Exercise 7-5: Inside Interactive Analysis

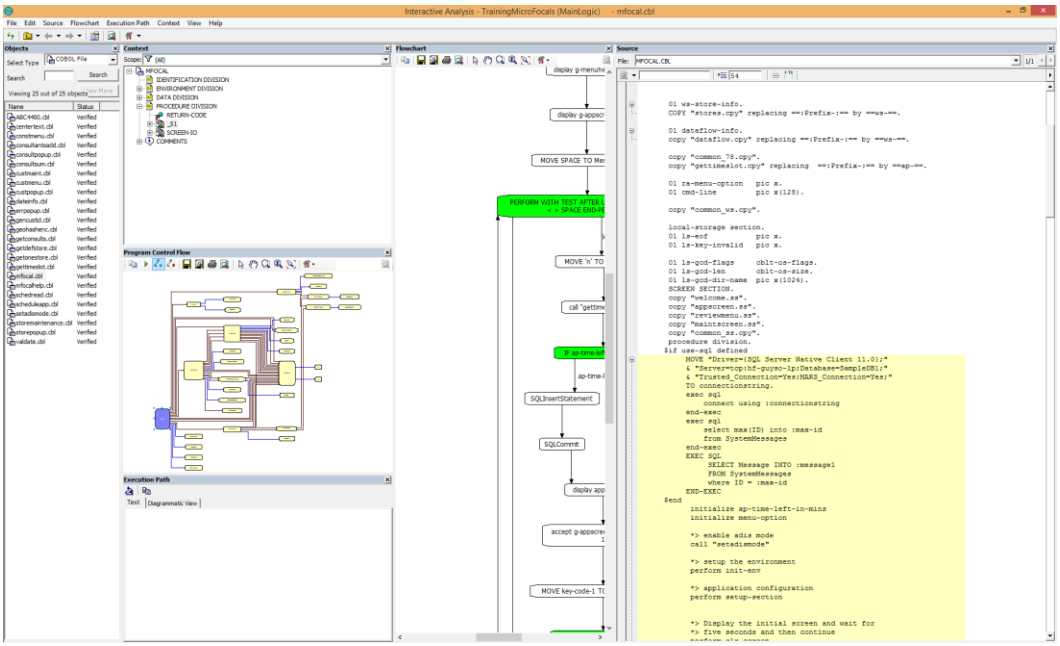
HyperView windows can be positioned according to your personal preferences. These positions are stored across CA sessions in your system registry settings, and will format the screen based on these values the next time you enter the HyperView screen.

Objective: You will use some of the HyperView tools to generate various views and practice moving the panes around the screen.

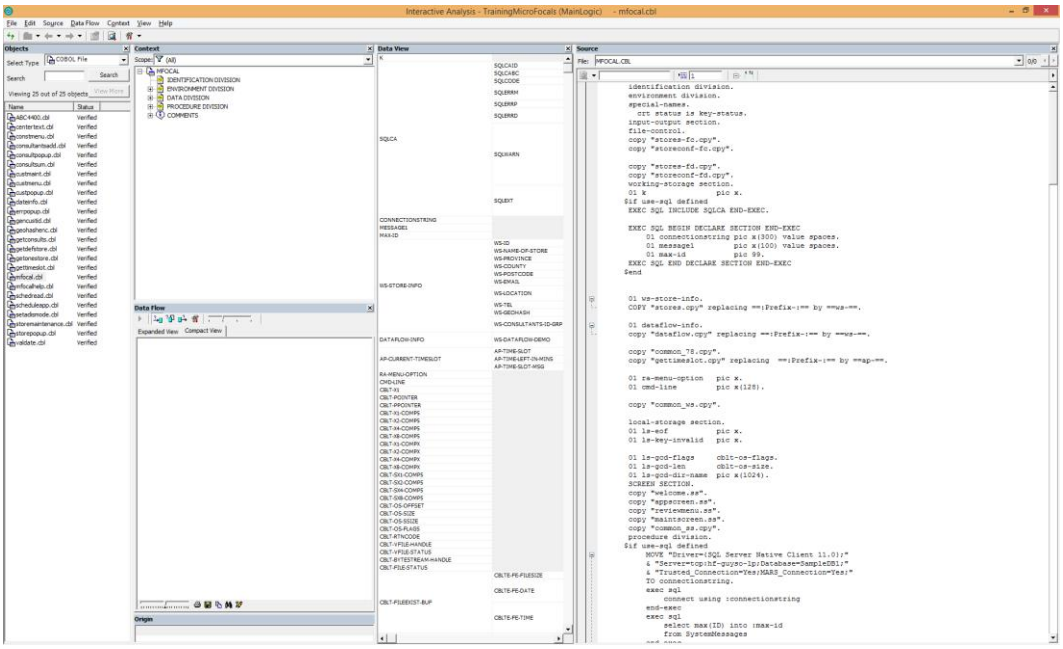
1. In the Repository browser, select the COBOL file **mfocal.cbl**.
2. Select **Analyze > Interactive Analysis**. This will start HyperView on **mfocal.cbl**.
3. Select **View > Classic Style**. You will see a screen similar to the following.



4. Select **View > Control Flow**. The Control Flow view shows the Objects, Source, Context, Program Control Flow, Execution Path, and Flowchart panes.

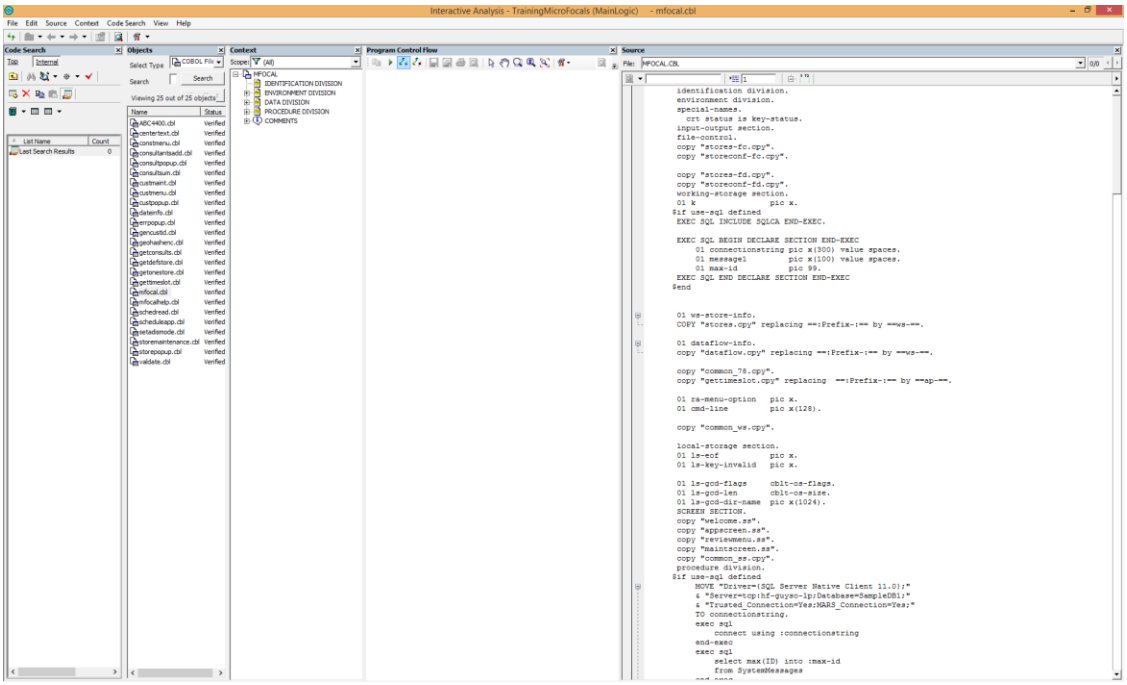


5. Select **View > Data Flow**. The Data Flow view shows the Objects, Source, Context, Data Flow (& Origin) and Data View panes.

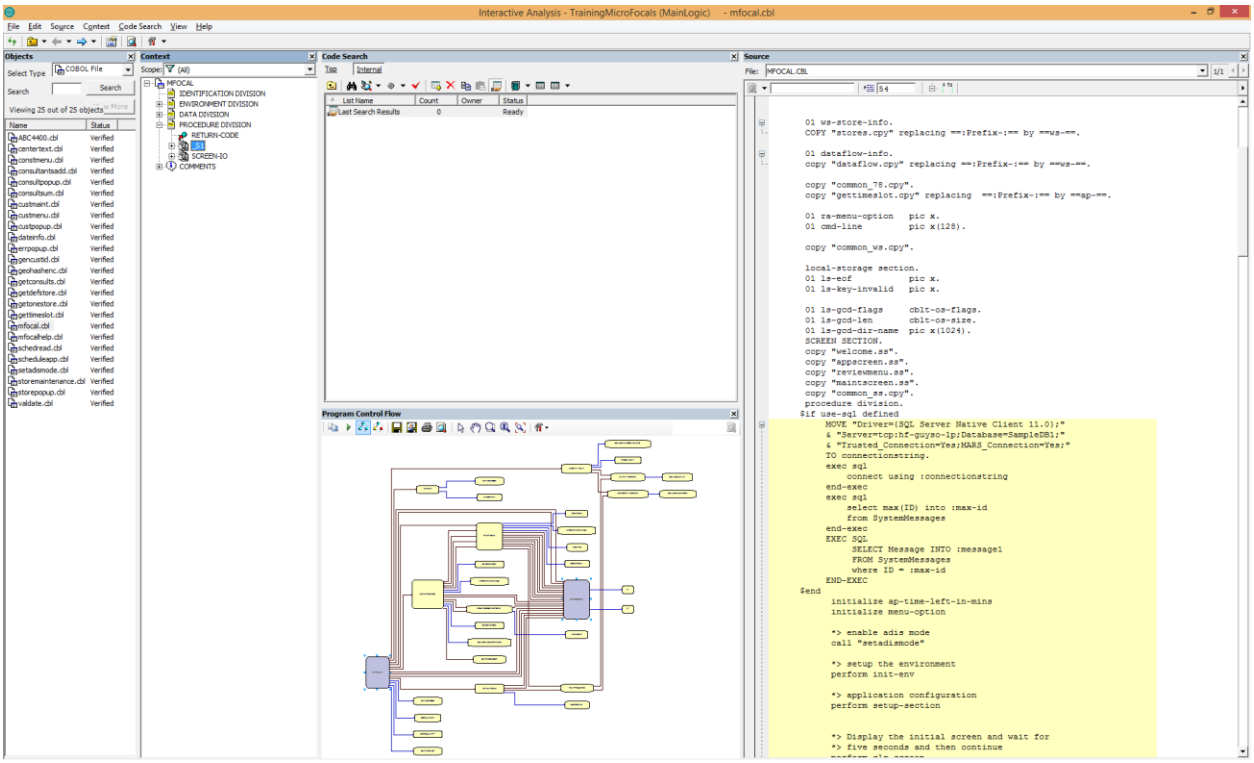


6. Select the following views to see what they look like:
    - **Classic Style**
    - **Program Control Flow**
    - **Code Search**

    Your layout may or may not look like the image below. Move the panes around to the desired position that best suits your needs.



7. Grab the BLUE bar at the top of the **Code Search** pane and drag it into the top of the **Program Control Flow** pane. As you drag the pane you will see horizontal or vertical dotted blue lines. These lines show where the dragged window will be positioned. Horizontal lines mean the window will be in the top or bottom. Vertical lines are for the right or left sides.
8. Resize the panes to your convenience.
9. Expand the procedure division in the context pane and select _S1 section



Exercise 7-6: Analyze Data Item Linkages

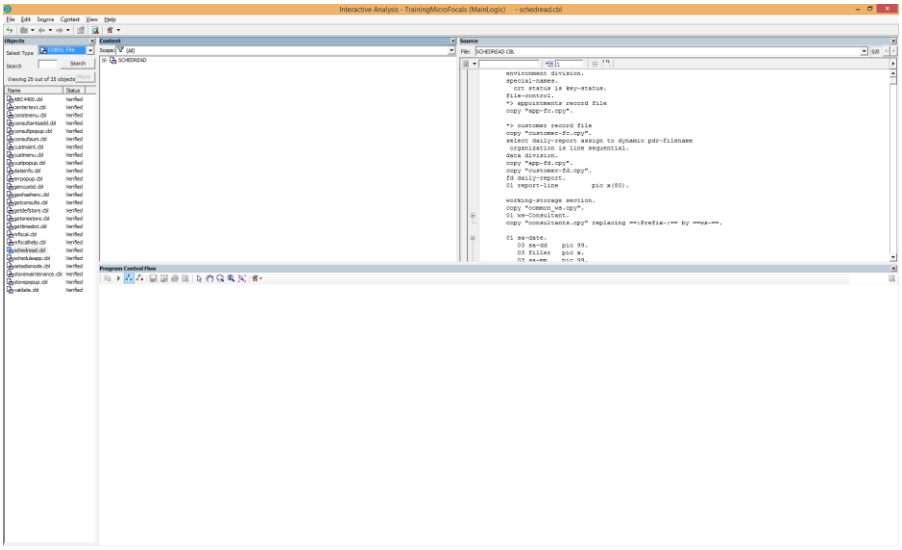
Within the Source and Context panes, you can build a “hierarchical drill-down” view into a legacy source file structure. You can select any code item or entity by clicking it directly in the text of the program. If the item you are selecting is of the lowest level in the code item hierarchy, then only this item’s line becomes highlighted in the color of the ‘Selected Construct Color’ as defined in User Preferences (default is pale yellow).

If it is an upper-level item, click in one of the fields that identify it: the paragraph heading for a paragraph, symbols IF or ELSE for an IF statement, and so on. However, if you selected a lower-level item, you can quickly navigate to its immediate parent or a more distant ancestor in the code as described below.

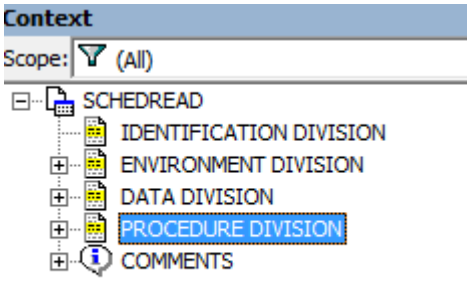
By expanding or contracting branches of a hierarchical (tree-like) structure, you can traverse the source quickly, navigating to or locating either declarative or procedural content.

Objective: You will inspect properties and attributes of various items within the Program Control Flow and Objects panes.

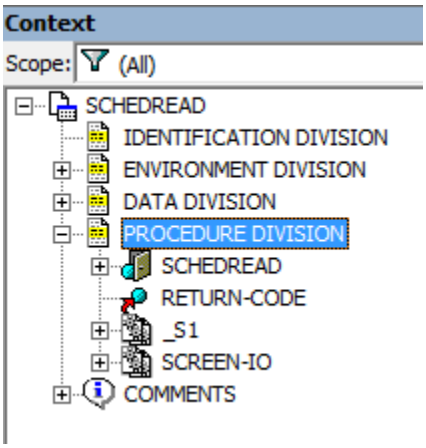
1. Inside Interactive Analysis, using the Objects pane (**View > Objects)** switch to **schedread.cbl**
2. Select **View > Classic Style**.
3. Select **View > Program Control Flow**.
4. Arrange the panes so that they look like the following image. **Program Control Flow** should be located at the bottom of the screen.



5. In the **Context** pane, click the **+** sign next to **SCHEDREAD** to expand the context tree, if necessary. Observe how the four COBOL divisions appear and the + sign next to SCHEDREAD turns into a - sign. The - sign indicates that to compress these “child” nodes back into the parent, you can left-click the - sign.



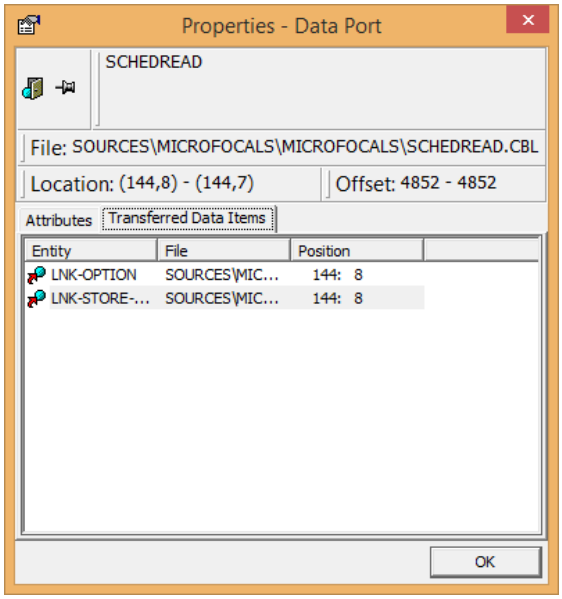
6. Click the **+** sign next to the **Procedure Division**.



You will see the following new child nodes:

- **SCHEDREAD** with a door icon and a blue circle represents a 'data port' – an item of data either entering or leaving this program
- **RETURN-CODE** is a special register for COBOL that is involved with program calls
- **_S1** is a default SECTION name, signaling that this program has no defined SECTIONS of its own
- **SCREEN-IO** is another SECTION name

7. Right-click on **SCHEDREAD** under **PROCEDURE DIVISION** and select **Properties**. You will see the **Properties** dialog box.

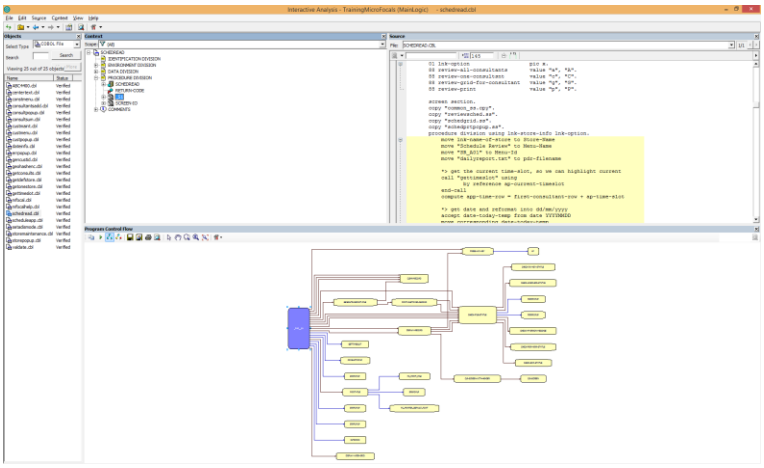


The **Transferred Data Items** tab will give you information about what is processed by this data port.

NOTE: In the **Attributes** tab, note the attribute named DEAD. This attribute will be set to TRUE only if the Hypercode entity is located in dead code and the **Perform Dead Code Analysis** option in the

Verification/Settings tab of **Workspace** options is checked when the file is verified. Many other important attributes are available in this tab as well.

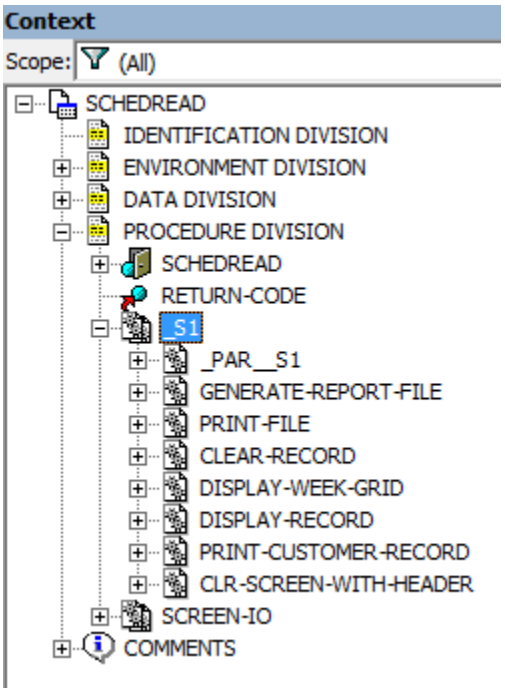
8. Click **OK**, and in the **Context** pane, select the **_S1** section. You will see a screen similar to the following.



Observe what happens in the Source and Program Control Flow panes. Any panes currently open in HyperView are synchronized based on your interactions with the pane in use.

The source pane is positioned at the start of the Procedure division — more precisely, at the start of the default section S1. Also, the selected source code, in this case the whole PROCEDURE DIVISON, is highlighted. Program Control Flow is positioned at the same point, showing subordinate paragraphs.

9. Double-click on the BLUE bar above the Program Control Flow pane to make the pane viewable in full-screen mode.
10. Double-click again to toggle back to the Hyperview window.
11. In the Context pane, click the **+** sign next to the **_S1** section.



Observe all the paragraph names that appear as subordinate nodes. If space becomes limited, you can resize the Context window by dragging the right or bottom borders between it and other panes. You can also scroll the window to reposition it over the data you require.

Also, notice the drop-down box at the top of the Context pane showing the displayed Scope. The default Scope is (All).

Explore the other Scopes (note that some of the listed items might be not present depending on the tool used):

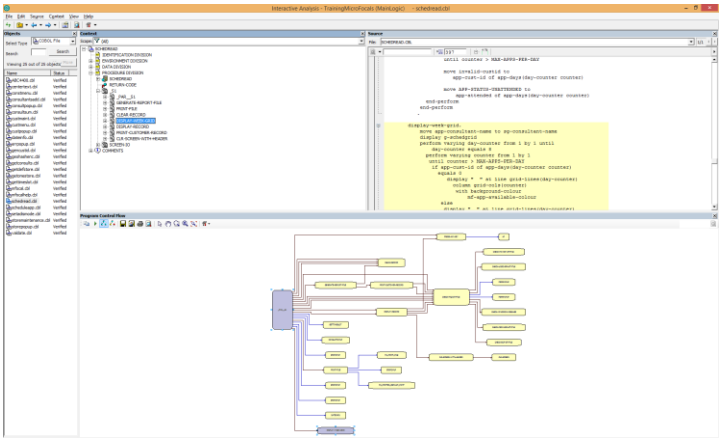
- **All**: displays all items.
- **Divisions, Sections and Paragraphs:** displays DIVISION, PARAGRAPH and SECTION objects and statements.
- **Data Declarations:** displays declarations in the DATA DIVISION.
- **Control Statements:** displays PORT, GOTO, PARAGRAPH, PERFORM, SECTION, STOP objects and statements.
- **Ports:** displays BLOCK, CONDITION, COMMON, PORT, DECLARATION, DIVI-SION, DDNAME, IF, PARAGRAPH, SEC-TION objects and statements.
- **Selected Code Search List:** There is also the possibility to restrict the tree to a specific set of constructs based on the results of an existing code search result. You can select a list in the subject program then the tree contains only items from the list and all its predecessors.

Exercise 7-6: Navigating Flow Relationships using Properties

Using Properties in HyperView, you can determine the flow of a given paragraph to the program.

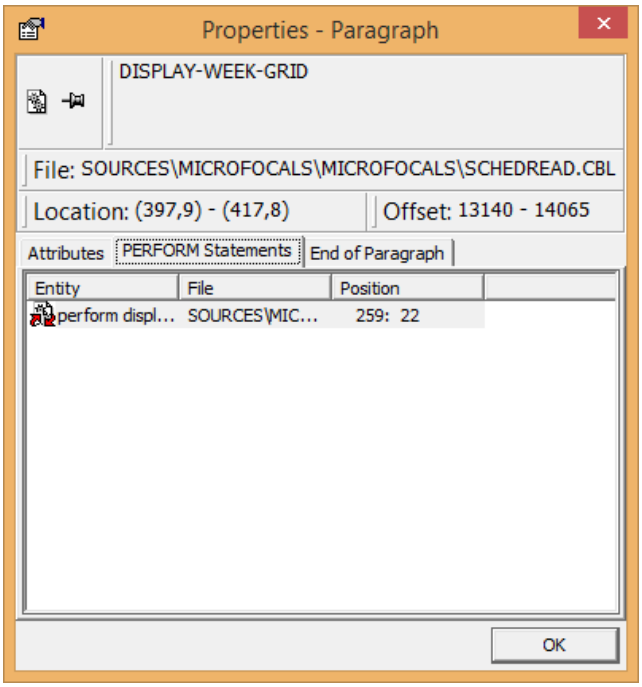
Objective: You will use the Properties dialog to navigate paragraph relationships. This exercise continues from the previous exercise. Still using the Source, Context, Objects and Program Control Flow panes and COBOL file schedread.cbl . Make sure the Context filter is set to All.

1. In the Context pane, click **DISPLAY-WEEK-GRID**. You will see a screen similar to the following.



   Observe how, once again, the Program Control Flow and Source panes are updated to reflect our position change within the code. You may need to scroll the Program Control Flow and Source panes to reposition their contents to enable you to more accurately see what has happened in these panes. Use the scroll bars for each pane to do this.

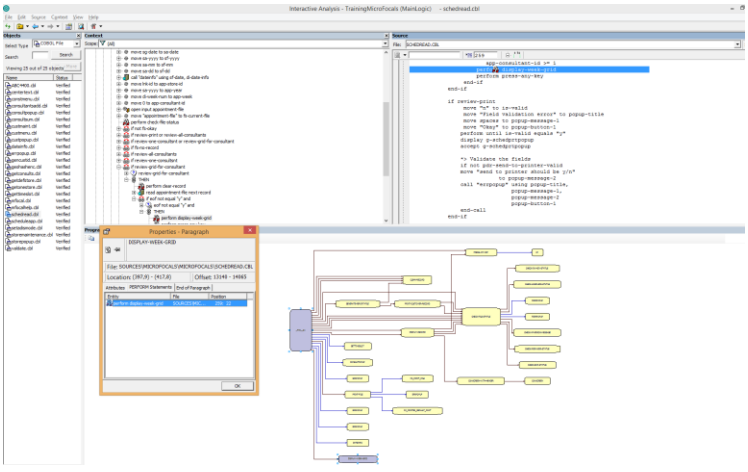
2. In the Context Pane, right-click on **DISPLAY-WEEK-GRID** and select **Properties**.
3. In the Properties window, select the **PERFORM Statements** tab.



The **Properties** window with the **PERFORM Statements** tab shows lines performing our chosen paragraph to illustrate:

- The selected paragraph is only performed from one location. No other lines of source execute this paragraph. We know this because there are no other entities displayed. There are no GOTOs, PERFORM THRUs, or paragraph fall-through entities.
- The contents of this Properties dialog differ from that for the previous (data port) because the Properties dialog is context-sensitive.

4. Select the entity **perform display-week-grid** in the Properties pane.



The Program Control Flow and Source panes are updated to show you the location of the PERFORM relationship selected in the Context Pane Properties dialog. The position in both the Program Control Flow and Source panes has been moved to the line of code that PERFORMS **display-week-grid**.

Also, the Context Pane has been automatically expanded to show the hierarchical relationship of this new source line — in other words, *its full parental history* in the Context Pane.

5. When you are finished, close the **Properties** dialog by clicking **OK**.

Exercise 7-7: Navigating Data and Program Relationships

With HyperView, you can follow the procedural flow between a paragraph and its PERFORM statements, as well as the flow between a data element declaration and its instances.

Objective: You will use the Properties dialog to navigate data and program relationships. This exercise continues from the previous exercise, using the Source, Context, Objects and Program Control Flow panes and COBOL file schedread.cbl. Make sure the Context filter is set to All.

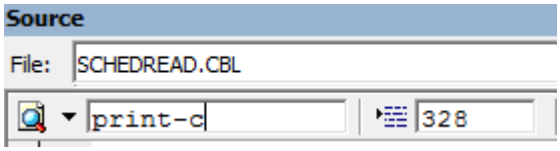
1. Select the **Source** pane.

   The **Source** pane provides a source-code view of the chosen legacy source file. However, Source pane provides many additional features for navigating and analyzing this source. Source pane also interacts with the other panes currently open in the HyperView window.

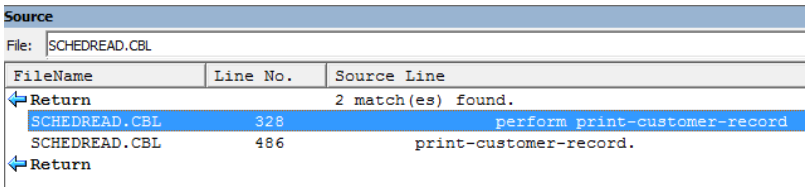
2. In the Source pane, type ***print-c*** in the **Find** box. In this example, you are looking for the print-customer-report paragraph.

   Observe that, as you type each digit, the search occurs instantaneously, updating both the Context and the Source pane positions with the first occurrence of the text in the Find box.

   In this instance, we have already found the first reference to the paragraph we are searching for. However, this is not the start of the paragraph, but the PERFORM of the paragraph.

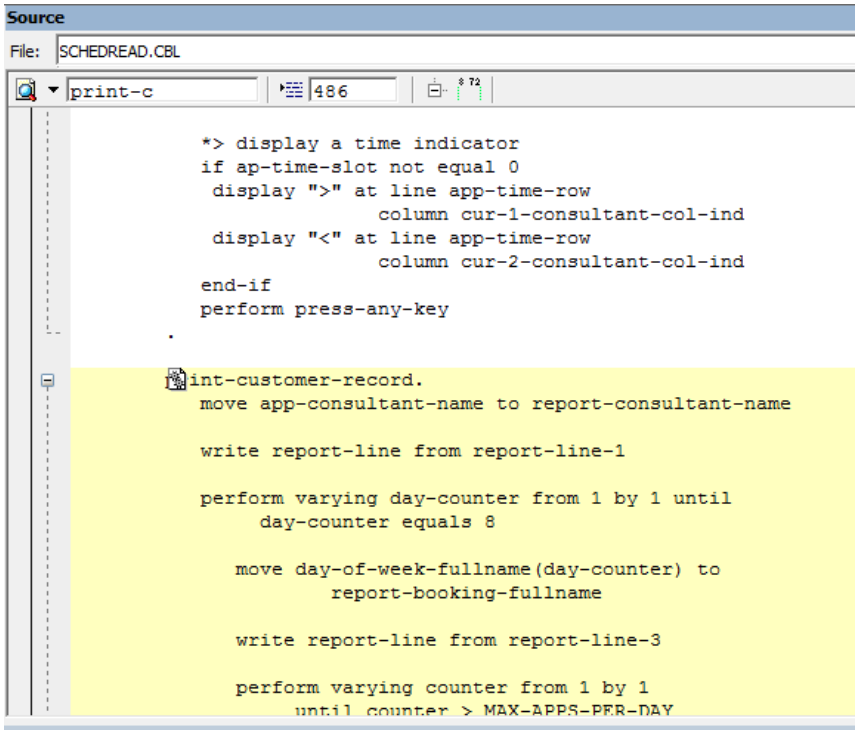


   Click the down arrow next to the  button and choose **Find All** in the drop-down menu.

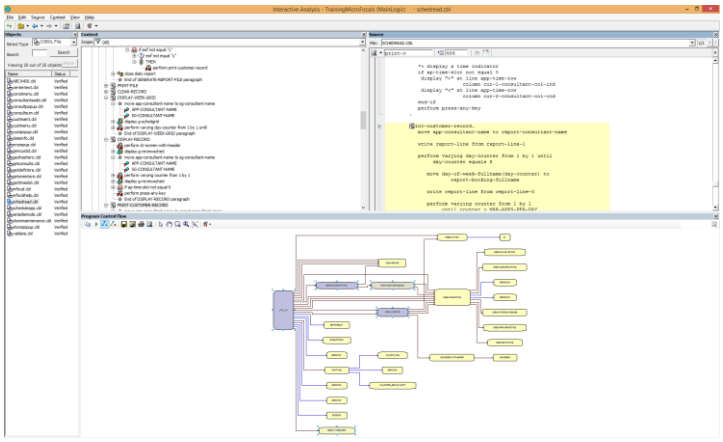


Observe the result of your search displaying every line with the search criteria print-c in it. You can see a list of all the results that occurred in this program including related copybooks.

3. Double-click the instance with line number **486** to go to that line of the source code.

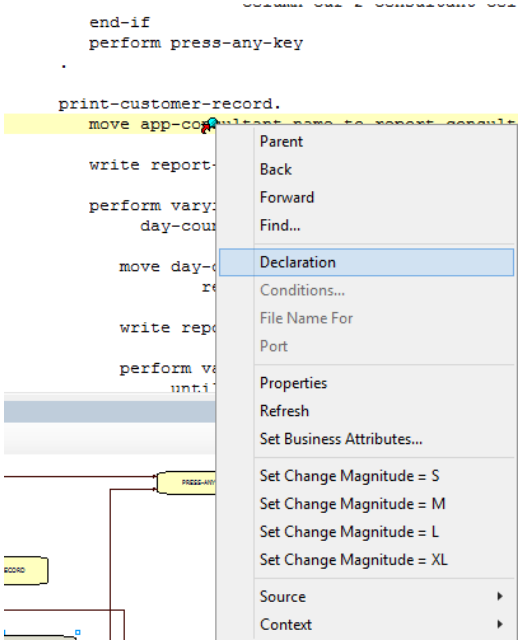


This is our desired location — the start of paragraph print-customer-report. Scroll down the Source pane to see the rest of this paragraph.



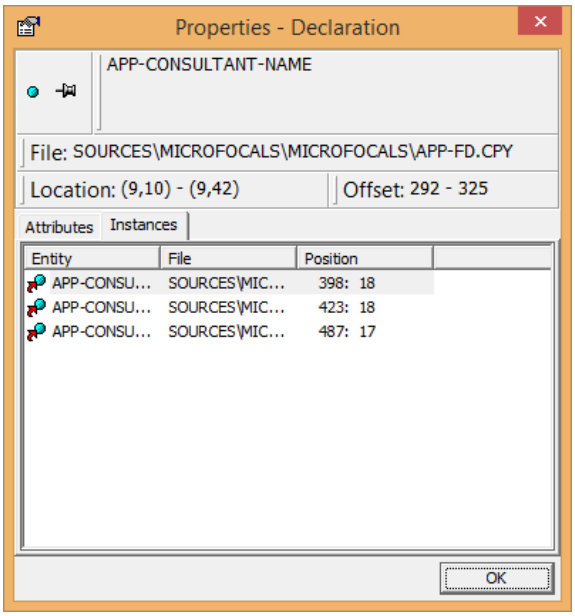
Observe how the other open panes (Context, and Program Control Flow) are updated with the new position you just selected in the Source pane.

4. Scroll down to the **move app-consultant-name to report-consultant-name** line of code (line 487).
5. Right-click on the variable **app-consultant-name** and choose **Declaration**.



The Source pane, as well as any other pane open in HyperView, will update itself to the location of the declaration of **app-consultant-name**.

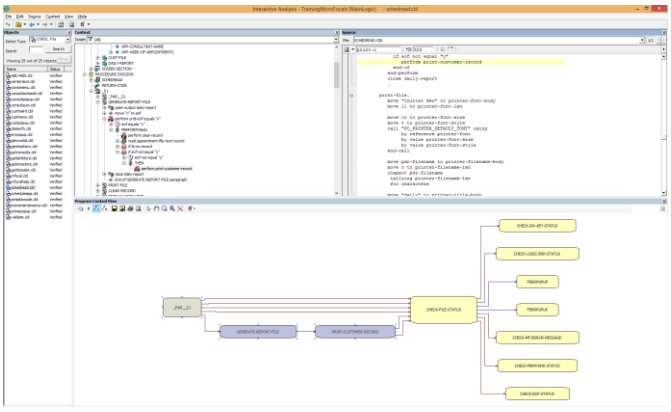
6. Right-click on the declaration and choose **Instances**. This will open a properties window in which you can see all instances of this variable throughout the program.



7. Click on the **Instances** tab. This displays all instances of the variable named in the tab at the top, along with file it exists in, and the corresponding location. Clicking on one of the results positions your source window cursor at the corresponding location.

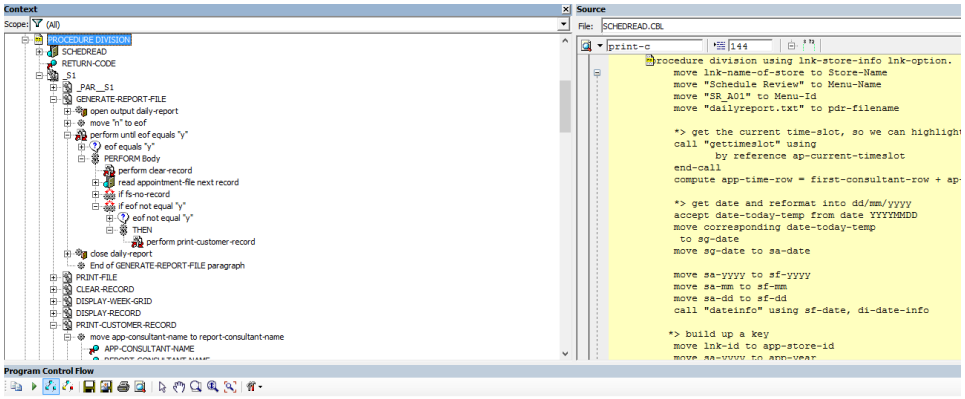
   The **Attributes** tab displays several important properties of the variable, such as (but not limited to) the Length, Picture, Normalized Usage, and Level.

8. When you are finished, press the **OK** button to close the **Properties** dialog.
9. Using the Source Pane (you can scroll up or use the Find method), locate and select paragraph **print-customer-record**.
10. Right-click on the paragraph name and select **PERFORM** statements. This will move you to the actual paragraph heading.



11. Right-click on the PERFORM statement and select **Called Paragraph**. Observe that we are now back at the **print-customer- record** paragraph heading.
12. In the Source pane, select the statement **move app-consultant-name to report-consultant-name** at line 487.

13. In the Context Pane, right-click the selected statement and select **Parent**. Keep doing this until you reach the top-level **PROCEDURE DIVISION**.



You should notice that you are traversing up the Context pane hierarchy. The parent of any statement in COBOL is its owning container:

- The MOVE is owned by the paragraph
- The paragraph is owned by the default _S1 SECTION
- The PROCEDURE DIVISION owns _S1

14. Right-click on **PROCEDURE DIVISION** and select **Back**. This allows you to trace back on any route that you had taken to get up to the current location.

    Another method is to use the  button to trace the route backwards like using a back button on a browser.

15. Click the  button on the HyperView toolbar repeatedly until you arrive at **move app-consultant-name to report-consultant-name** at line 487.
16. . Observe that you traced the exact same route in reverse.

    You can also retrace the route forward using the  button or Forward from the Right-Click menu.

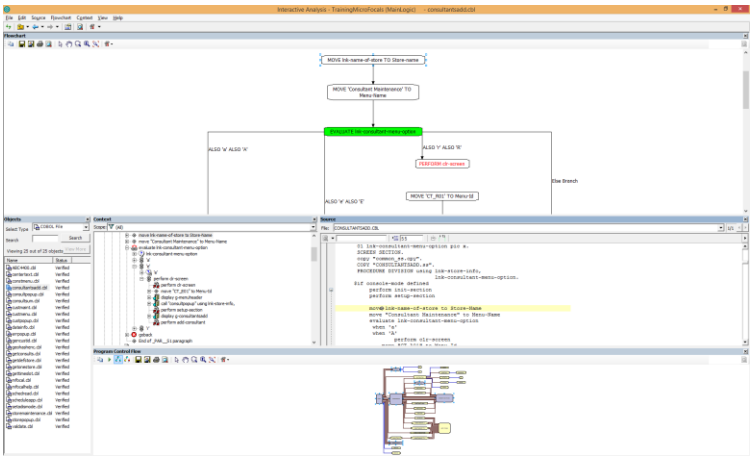
Exercise 7-8: Using the Flowchart and Animator

Flowcharts are a great tool for understanding business logic. A flowchart shows medium-level entities (statements) and thus provides a rougher picture of the paragraph than the corresponding part of the parse tree.

This intermediate level of detail, in combination with showing operation flows, provides an “algorithm-oriented” representation of the program. Such a representation can be especially helpful for a programmer who navigates through the legacy code.

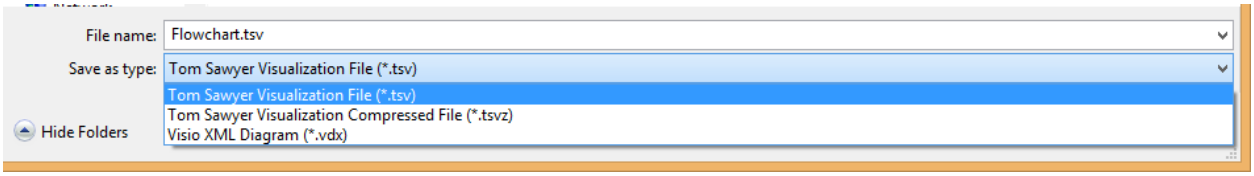
Objective: You will use the flowchart and animator to navigate your code. This exercise continues from the previous exercise, using the Source, Context, Objects and Program Control Flow panes and COBOL file consultanstadd.cbl. Make sure the Context filter is set to All.

1. Go to line 55.
2. Select **View > Flowchart** to open the Flowchart pane.



Observe the flowchart drawn to represent this paragraph. The statements in the Flowchart pane are drawn to aid your understanding of the logic flow. This pane is mainly used as a "passive" pane, but clicking on its elements provides the same synchronization functionality as other panes.

3. Select **Flowchart > Save Diagram As…** but do not save the flowchart.



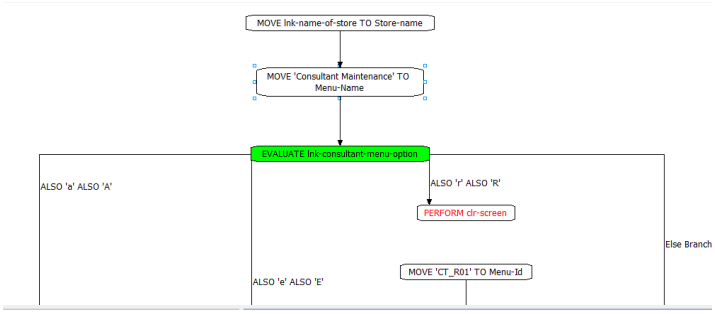
We can use the Save function to store the diagram as any valid flowcharting format for which we have software on the machine. Tom Sawyer is a third-party tool used for generating diagrams in CA. Do NOT save in this format. If you have Visio or Visio reader installed, you may save in .vsd or .vdx respectively.

However, do not save the flowchart.

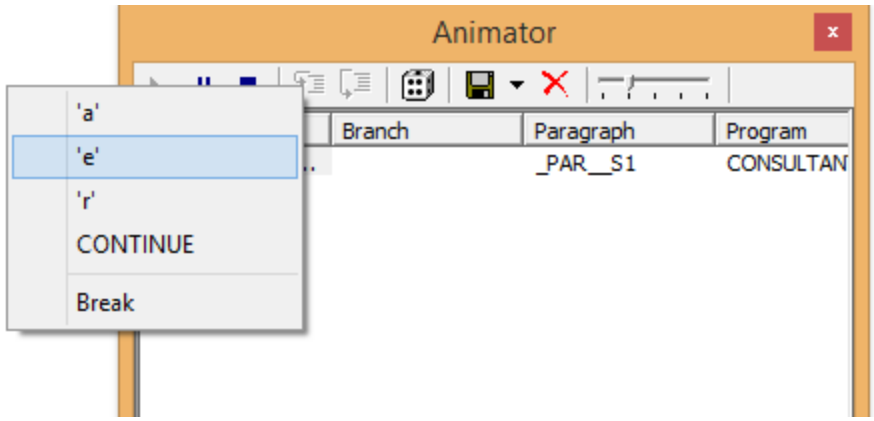
4. Click **Cancel**.
5. Select **View > Animator**. The **Animator Pane** allows you to trace the path that a piece of code follows during execution.
6. Push the  **Step Into** button on the Animator Pane. This allows you to step through code one step at a time.

```
PROCEDURE DIVISION using lnk-store-info,
                         lnk-consultant-menu-option.
$if console-mode defined
     perform init-section
     perform setup-section

     move lnk-name-of-store to Store-Name
     move "Consultant Maintenance" to Menu-Name
     evaluate lnk-consultant-menu-option
      when 'a'
      when 'A'
           perform clr-screen
        move "CT_A01" to Menu-Id
           display g-menuheader
           perform add-consultant
           perform setup-section
```



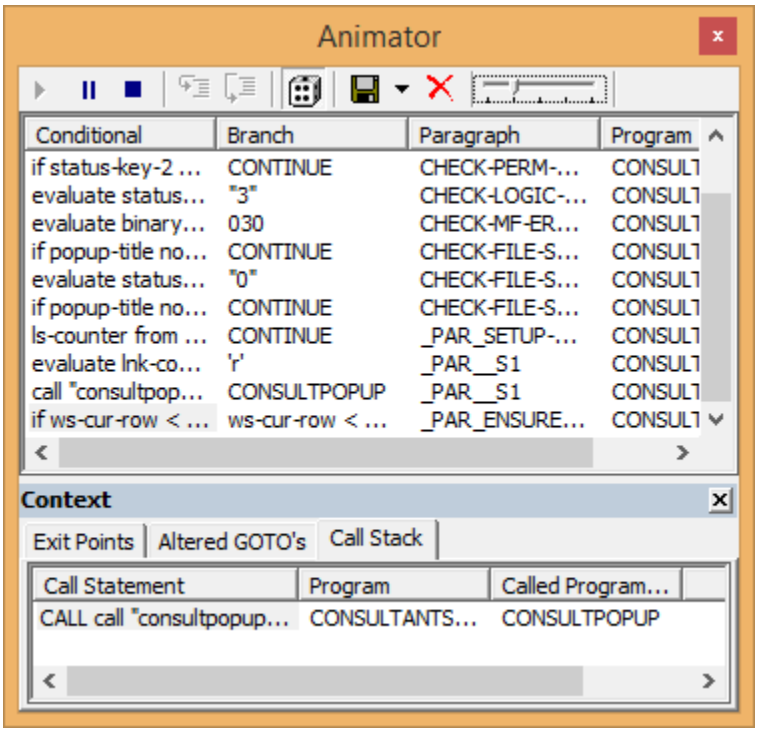
7. Push the **Play** button on the **Animator** Pane. The **Flowchart** pane follows the flow while the animator steps through the code.



This will step through the code entity by entity until it reaches an IF statement where it will stop and ask you what path to take. The assumption is that you "know" the data that will be in the test case and will be able to use valid data conditions to follow the flow.

8. Toggle the button to set **Automatic Random Choice** mode and push the **Play** button.

The **Random If** choice button toggles between automatic and manual mode for choosing "If" paths. The Animator will continue running the code, selecting its own path randomly when it reaches an IF statement.



The Animator pane shows a record of the various IF branches.

- Use the **Slider** control to vary the speed of the Animator
- Use the **Save** button to store the results as a list that can be used by other tools, such as Code Search or Context

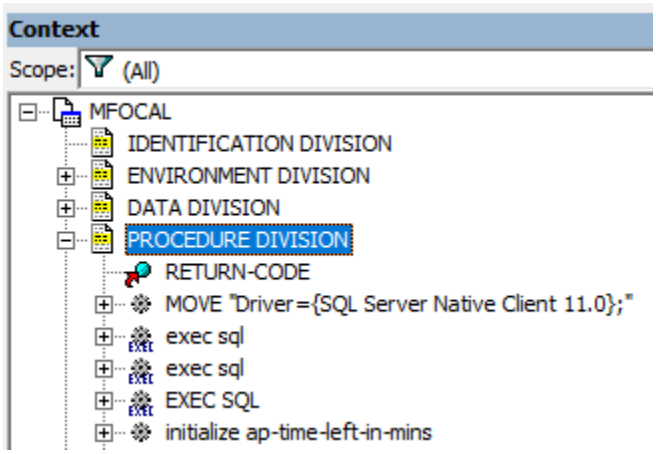
9. Push the **Stop** button to stop Animator.
10. Navigate the different entries in the list and examine the provided information in the different panes.
11. Close Animator.

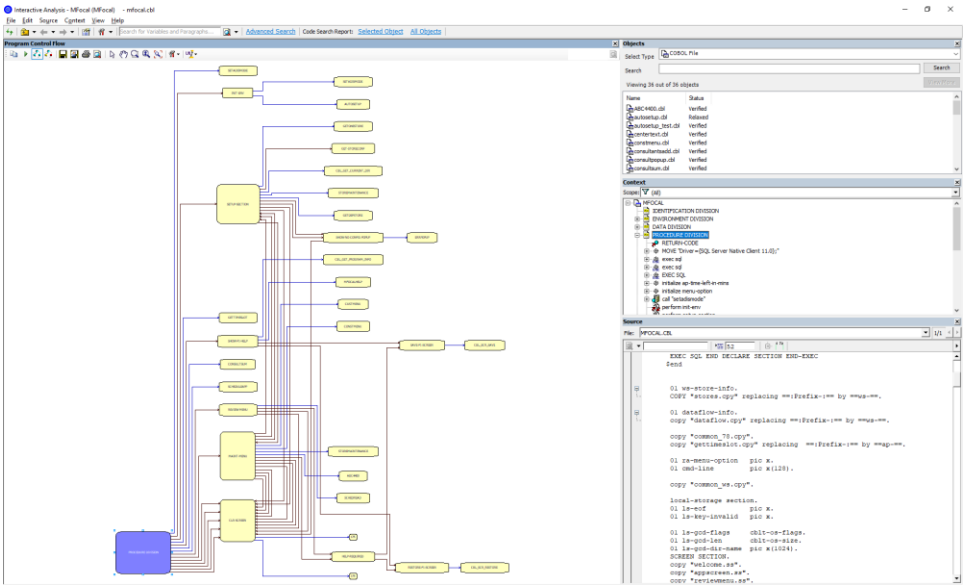
Exercise 7-9: Using Program Control Flow to Analyze Call Relationships

The Program Control Flow pane is a diagrammatic representation of the overall flow of a program. Like all HyperView panes, it interacts with other panes. It provides two different "modes" or representations of this flow, with many options to control their appearance and the ability to export diagrams for subsequent manual modifications.

Objective: You will use the Program Control Flow pane to graphically trace the execution flow across paragraphs in a program.

1. In the Objects pane, select the COBOL file **mfocal.cbl.**
2. Make sure that only the **Source, Context**, **Objects** and **Program Control Flow** panes are selected.
3. Close all other open panes.
4. In the Context pane, navigate to the **PROCEDURE DIVISION** and expand the node





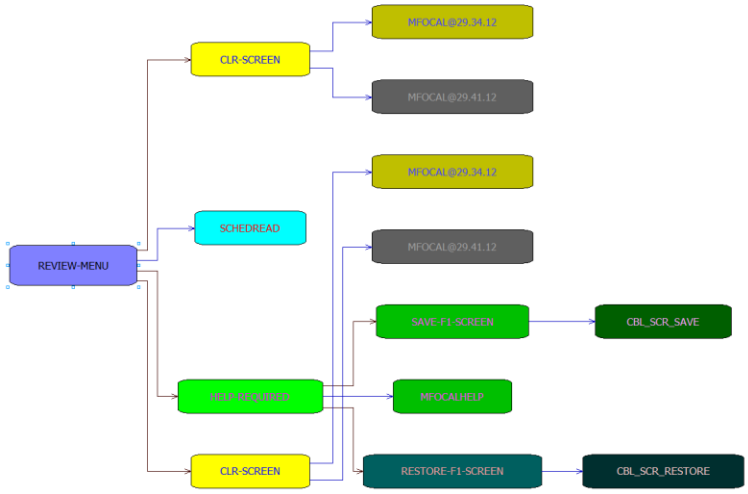
The Program Control Flow pane now contains the full program structure. Observe the following:

- Because you have selected the procedure division, Program Control Flow shows the whole program structure.
- Colored boxes represent paragraphs and called programs.
- Procedure Division is blue and has a shadow because it is the first paragraph and is currently the selected paragraph.
- Lines between boxes represent program logic flow as follows:
    - **Black** = PERFORM
    - **Green** = GO TO
    - **Red** = Fall Thru
    - **Blue** = transfer control to an external node (e.g., a CALL statement)

In Program Control Flow, you have the following diagramming tools available:

- **Select:** this tool allows you to select an object in the diagram. The selection becomes the focus in the diagram and in all other panes in HyperView.
- **Hand**: this tool allows you to manipulate the diagram position by click-and-drag
- **Marquee Zoom**: this tool allows you to select then draw out a selection in the diagram space. On releasing the mouse button the diagram shows only the selected area.
- **Interactive Zoom:** this tool allows you to select an area of the diagram to be the focus of a zoom in and zoom out by moving the mouse down or up (respectively) in the diagram space
- **Fit in Window**: this tool allows you to shrink or expand the diagram to get a best fit in the available diagram space

5. In Program Control Flow, select the **REVIEW-MENU** paragraph.
6. Using the Sub-Graph ( ) and Sub-Tree ( ) modes, switch between the two and observe the differences between the two representations of the control transfer diagrams.



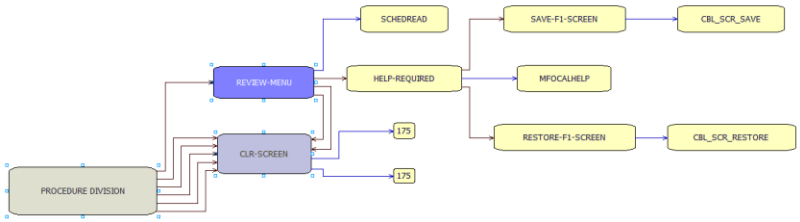
Why? We need to emphasize the crucial benefits of the SubGraph view. Sub-tree provides a “true” picture of the control transfers – with all their complexities. SubGraph “simplifies” this view by showing each paragraph only once.

SubTree mode: In the SubTree representation **Program Control Flow unfolds the graph and displays it as a tree of paragraph instances**. With each paragraph call having a unique copy, you can distinguish all call consequences that could occur in the program.

The SubTree Mode provides a full, complex diagram, with *all control transfers*. Note how the hierarchical layout and automated coloring "draws the eye" to emphasize controlling paragraphs (in blue) and call-depths (green and yellow in this case).

7. Select **Program Control Flow > SubGraph Mode**.

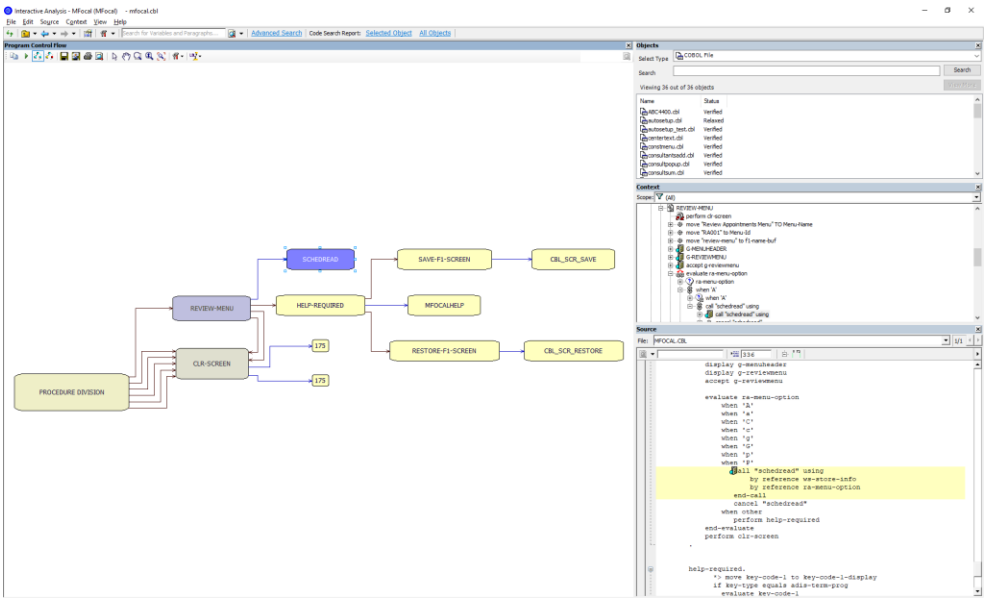
   In the SubGraph representation, a paragraph is depicted ***by one rectangle and one rectangle only***. This "graph" representation depicts a high-level view of transfers of control and thus is critical to gaining a comprehensive view of the overall program functionality without getting lost in the intricacies of repeated executions of the same piece of code.



   The SubGraph Mode represents the same code and control transfers as the SubTree Mode, but in an important different way – it mathematically "folds" the control flows to show a condensed version – where each paragraph appears just once.

8. In Program Control Flow SubGraph mode, left-click and select the paragraph **SCHEDREAD.**

   Why? We are going to track down a "perform thread" to follow a process path.



Notice the following:

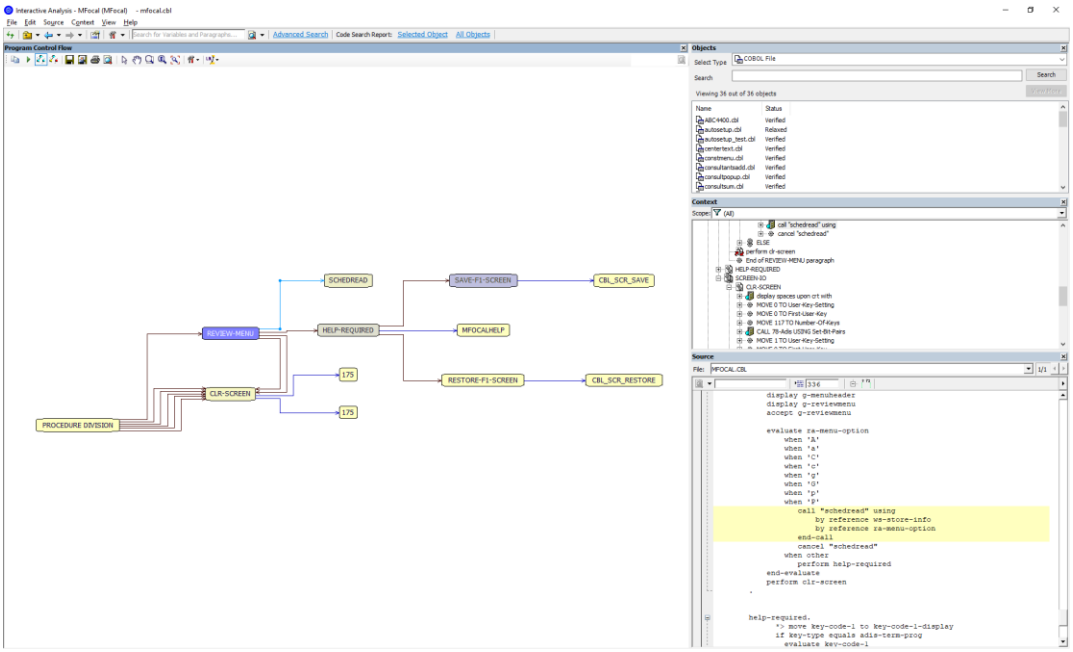
- The newly selected paragraph turns blue and has handles around it to signify selection
- The previous selection is now a lighter blue – to signify that this is where you came from
- Other Panes are updated too. See the Context Pane and Source Pane changing as you make selections in Program Control Flow

9. Continue this left-click paragraph selection process through **HELP-REQUIRED** and **SAVE-F1-SCREEN.**

   If you cannot see all these paragraphs in the same diagram, open the **Project Options**  and go to the **Program Control Flow** tab. Adjust the **Call** and **Caller Depth** levels to **5** (this should be unnecessary if using the default settings).

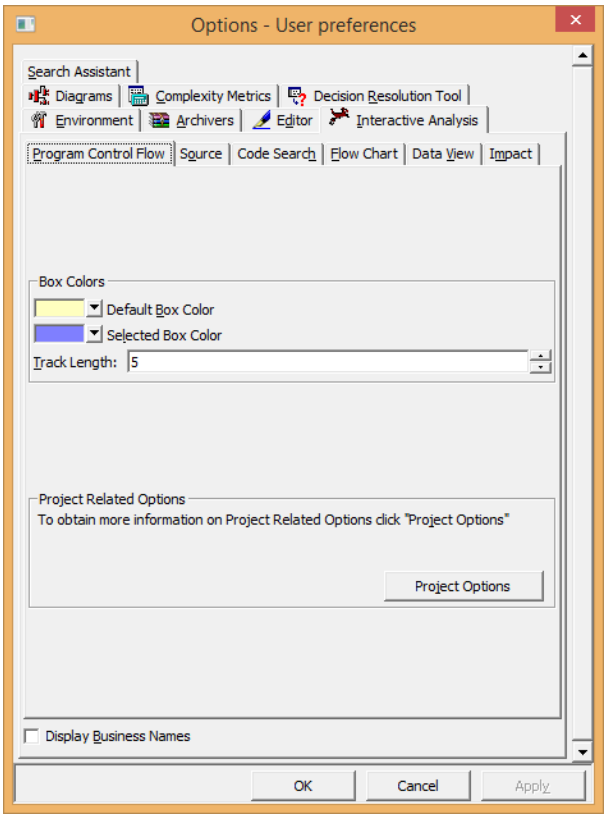
   Notice that each previously selected paragraph is a gradually lighter shade as you move down the "perform chain."

10. Select the **line (edge)** between **REVIEW-MENU** and **SCHEDREAD.**



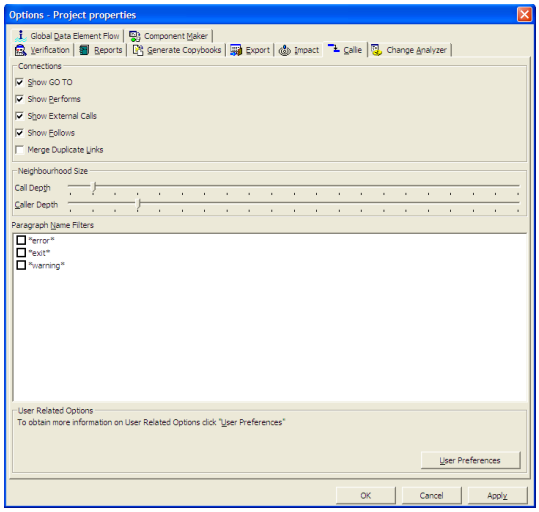
   Notice how the PERFORM statement is highlighted in the other open panes. Each line in Program Control Flow can also be selected to understand how the paragraph is reached. Other panes are also updated as a result.

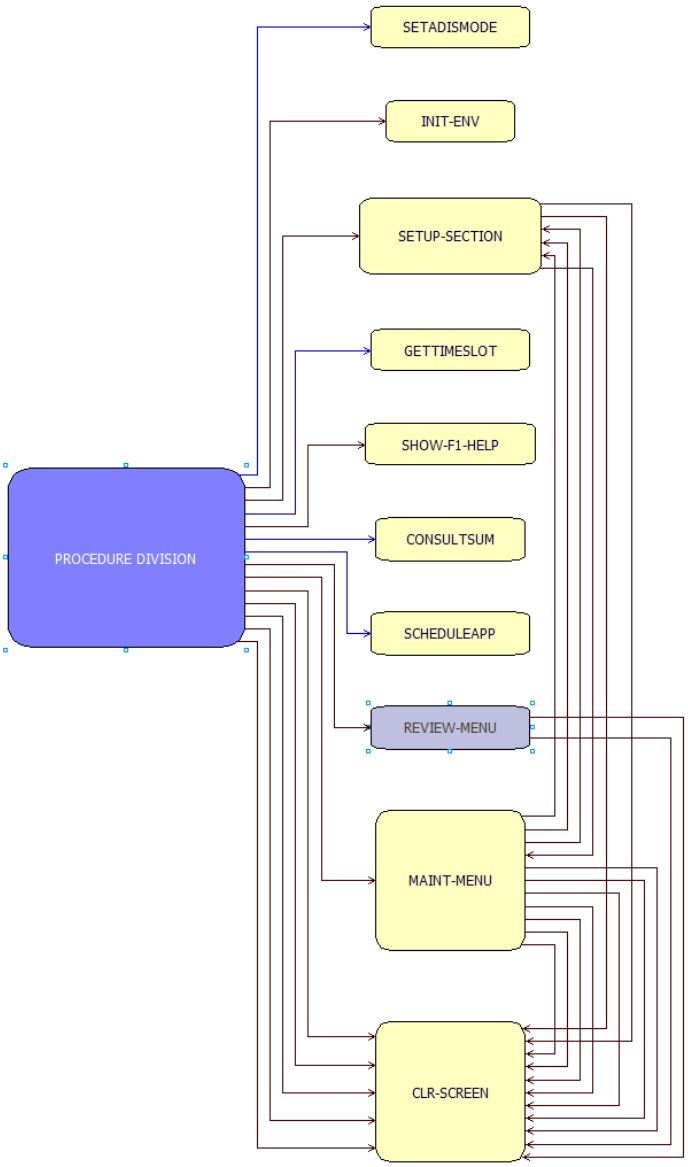
11. Select **Procedure Division** in SubGraph mode.
12. Click the down arrow next to the HyperView Options icon on the main tool bar () and select **User Preferences.**
13. Select the **Interactive Analysis** tab and then select the **Program Control Flow** tab.



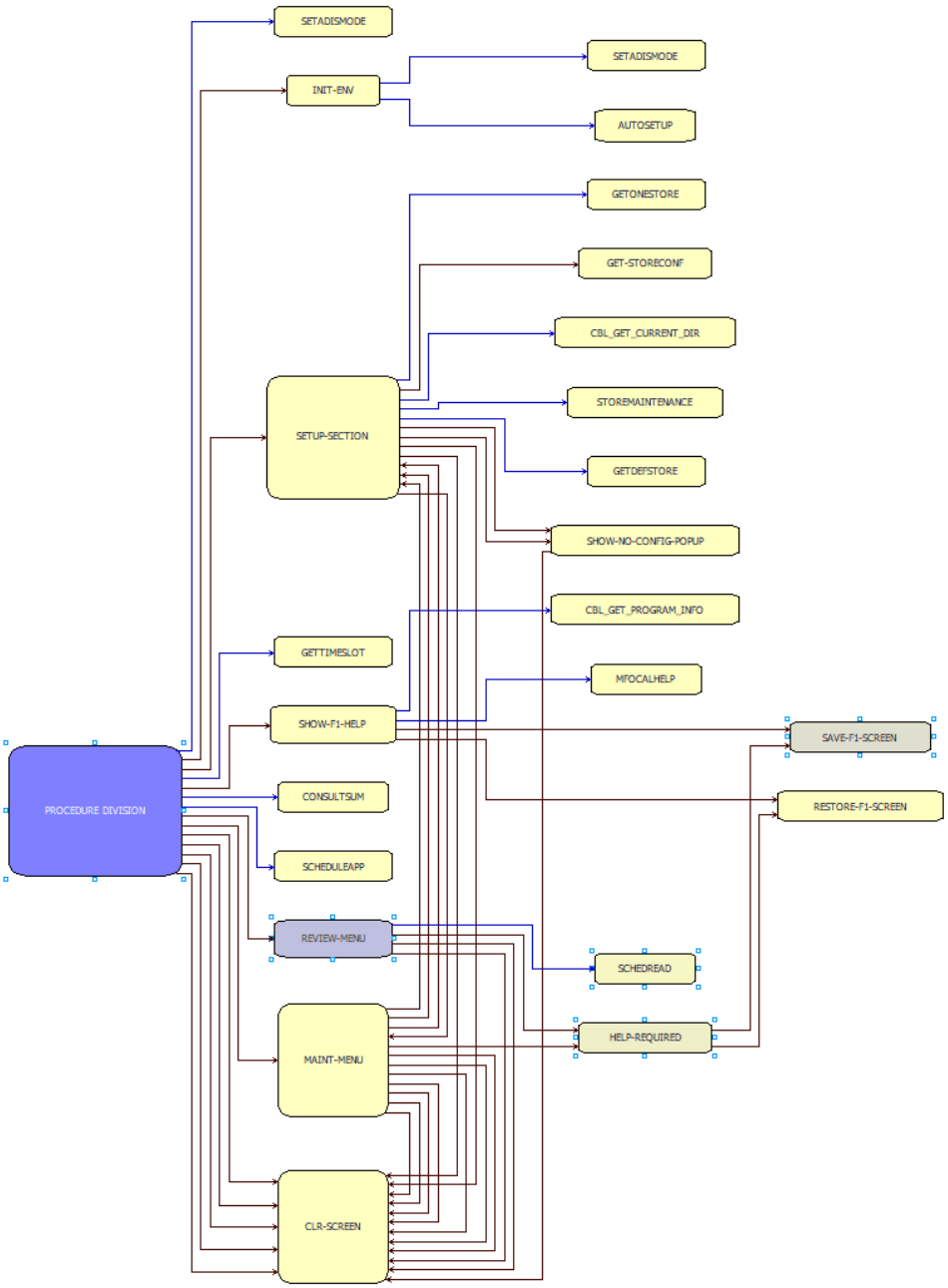
Box colors: determines starting and selected colors. Track length determines the color-shaded "ancestry" when following an execution path as described above – where color shades change as you navigate down a path.

14. Click **Project Options.** All the diagrams drawn above were drawn with Call Depth and Caller Depth set to **3**.
15. Re-set **Call Depth** to 1 and click **Apply**. Observe the impact on the SubGraph mode.





16. Re-set **Call Depth** to 2 and click **OK**. Observe the impact on the SubGraph mode. As you can see – increasing detail is obtained.



17. Make sure **Call Depth** and **Caller Depth** are reset to the default **3** and press **OK**. This will reset the project back to defaults.
18. Close the Program Control Flow pane by clicking the **X** in the upper right-hand corner of the pane.

Using code searches and code search reports

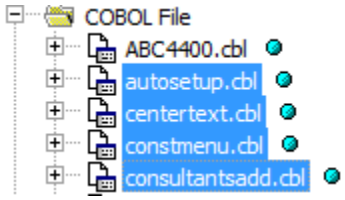
Code searches allow you to quickly perform complex searches for code constructs, coding standard violations, possible performance issues and defacts in the code.

You can run the searches using existing built in code searches or create your own specific code searches (for example, creating code searches library to match your organization coding standard rules)

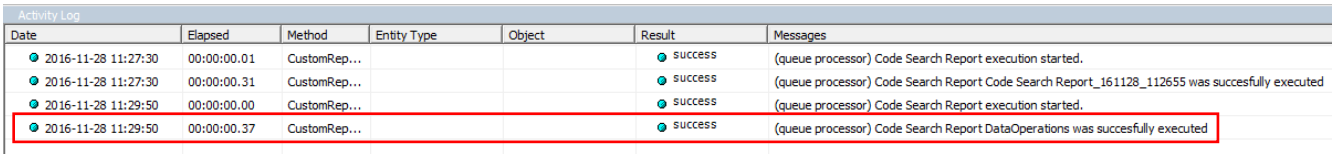
You can run a code search or multiple code searches over your entire project or a selected set of files and you can generate an HTML report to expose the findings across the organization.

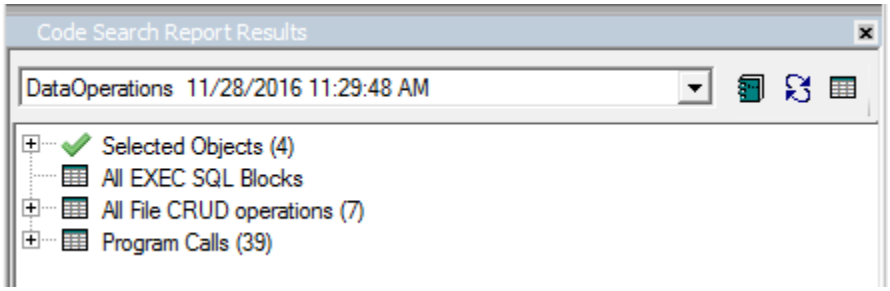
Exercise 7-10: Basic code searches

1. From the main tool window open the **MainLogic** project and expand the COBOL files
2. Multiselect the first 4 files after ABC4400.cbl (using the standard CTRL + Left Click or Shift + Left Click)



3. Right-click and select Code Search Reports→ Code Search Reports…
4. Expand COBOL→Data Ports and continue expanding to find and check:
    a. Program calls
    b. All file CRUD operations
    c. All EXEC SQL blocks
5. Name the report: DataOperations
6. Click Run to run the report
7. When the report is ready, a notification will be sent to the activity log. Double clicking it will open the Code Search Report results pane:

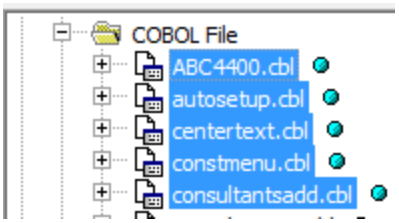




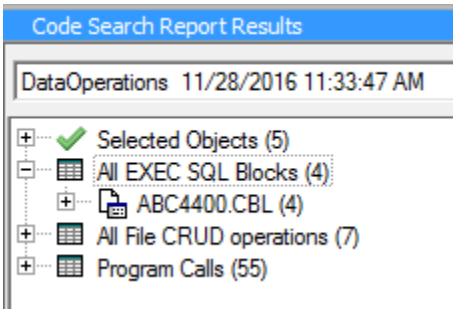
8. Selected Objects shows which files have been included in the search, and the rest of the entries are results.
9. Expand the results and double click the results tree leaves to see the line in the code.

Notice that you can right-click the result or source view to open the interactive analysis window for that file.

10. Add ABC4400.cbl to the selected file by CTRL+Left Clicking it



11. Right-click on one of the selected files and select Code Search Reports→DataOperations
12. The report we be re-executed on the 5 selected files, and the results will include the EXEC SQL blocks

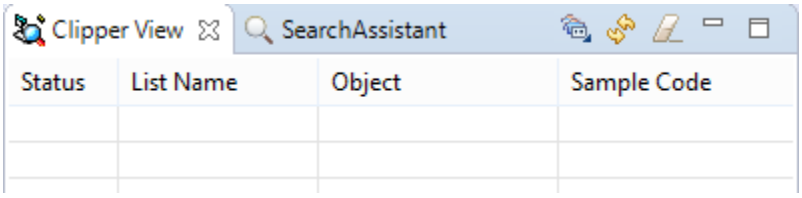


13. Right click on the **MainLogic** project and select the DataOperations custom report to run the code searches for all the project files
14. In the Custom Report Results toolbar, click the create reports button () to see the HTML report.

Exercise 7-11: Exporting results to Visual COBOL for Eclipse

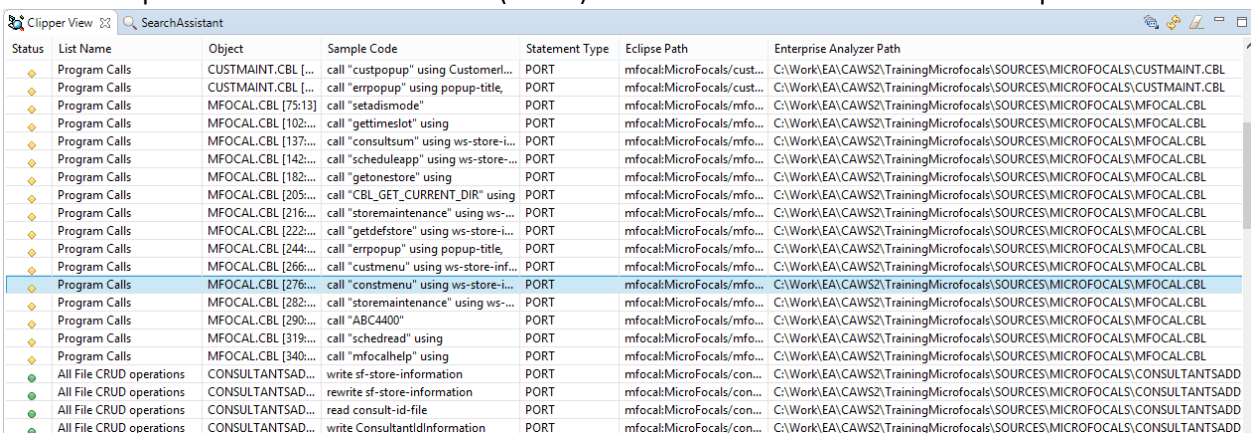
In order to import code searches results (POIs) from COBOL Analyzer to Visual COBOL, you'll first need to install the Eclipse plugin. The instruction can be found under <COBOL Analyzer installation folder>\docs (Default is: C:\Program Files (x86)\Micro Focus\COBOL Analyzer\Docs).
File name is: Installing EA Plugins for Eclipse.pdf

1. Using the results from the previous exercise, click the Export to Eclipse Plugin button ()
2. Save the file.
3. Open Visual COBOL for Eclipse and load the training material project.
4. Make sure you see the Clipper View pane in Eclipse:



5. If you don't see it, go to Windows->Show View -> Others->Micro Focus->Clipper view (In Eclipse)

6. Click the Import code search list from file ( ) and browse to the saved file from step 2:



7. Clicking on the different entries will take you to the found line in the code
8. Once you made code changes and saved the files, you can go back to COBOL Analyzer and rerun the report ( )

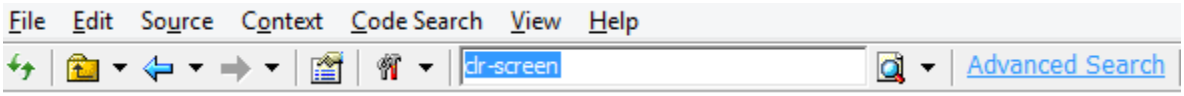
You should make sure the files are updated in order to receive the updated results, you can use the source synchronization mechanism to achieve that or manually refresh the changed sources and re-verify.

Exercise 7-12: Running quick searches

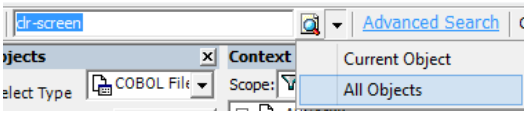
The following exercise will show you how to run quick searches with Interactive Analysis Search facility for the active program. Quick search will find variable's declarations, usages and paragraphs. You will start with MFOCAL.CBL in Interactive Analysis.

You will be using the **Source, Context, Objects**, and **Code Search** panes and will load a new file, MFOCAL.CBL**,** in the source pane.

1. Type *clr-screen* into the quick search box



2. Press **Enter** or click the icon to the right of the text box to find results in active program
3. Review the results in the code search pane, you can navigate the results by clicking them
4. Click the dropdown button to change the search scope to all the objects in the project



5. Review the results
6. Return to mfocal.cbl using the **Objects** pane
7. Quick search also accepts wildcards, type **-menu* and run the search
8. Notice that the results now contains paragraphs as well as variables declarations and usages

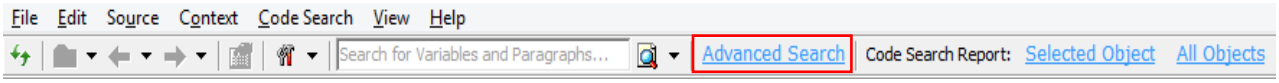
Exercise 7-13: Advanced code searching – Creating new code searches

The following exercise will familiarize you with Interactive Analysis Search facility for the active program called Find. The Search facility capabilities are comprehensive, so we will restrict ourselves to some common search examples as illustrations of what can be achieved. You will start with MFOCAL.CBL in Interactive Analysis and search for all EXEC SQL Statements.

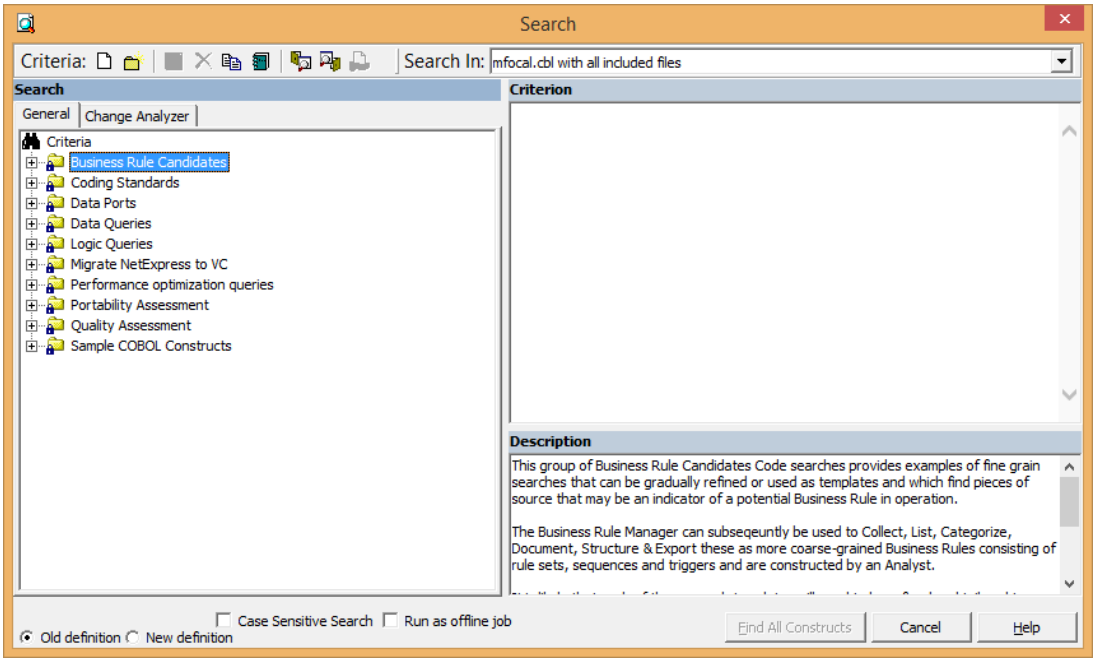
You will be using the **Source, Context, Objects**, and **Code Search** panes and will load a new file, MFOCAL.CBL**,** in the source pane.

Objective: You will use Find with the Objects pane to run a search.

1. Click on the **Advanced Search** hyperlink (the Search facility for a single program) in the Interactive Analysis Toolbar at the top of the screen. This will start the Search facility for the active program.



This uses the same Queries available from the Code Search pane and the Custom Report feature pane, but will only apply them ONLY to the selected file. This is a good way to quickly test a query.



Notice the **Search In:** text box indicates that you are searching **MFOCAL.cbl** with all included files. Observe the same existing pre-built advanced searches that you saw when using Code Search against the entire project. Try selecting each one from the list on the left to see its definition on the right.

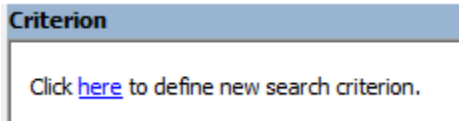
2. Ensure the top level (Criteria) is selected in the Search pane. Selecting Criteria ensures that the new Code Search query will be located at the top level.

Now you will create a new Code Search query designed to find EXEC SQL statements.

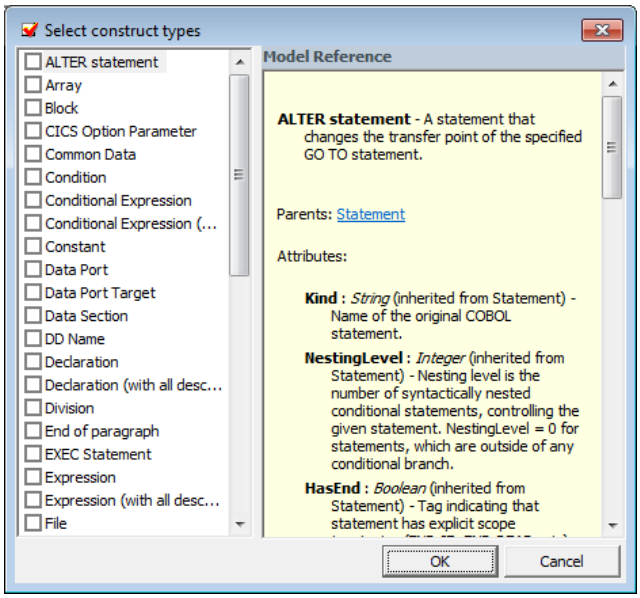
3. Click 'New Criterion' ( ) to create a new query.

NOTE: Be sure to always select New Criterion before creating a new query.

4. Enter a suitable name (such as "EXEC SQLs") and press **OK**.
5. Click on the blue 'here' as shown below to start constructing your query.



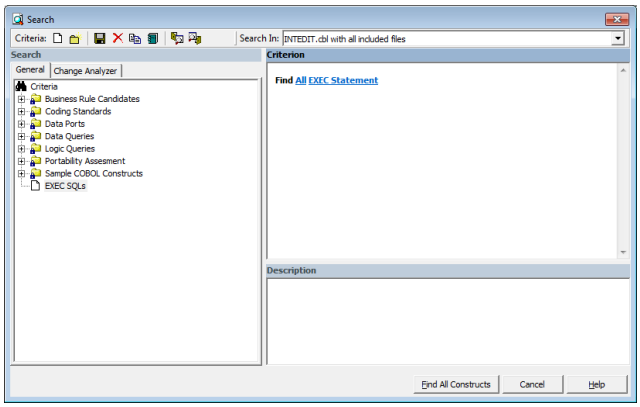
You will see the **Select Construct Types** dialog box.



The first step is to choose the construct type that you want to find. These do not always resemble COBOL constructs – CA's repository was designed for several languages.

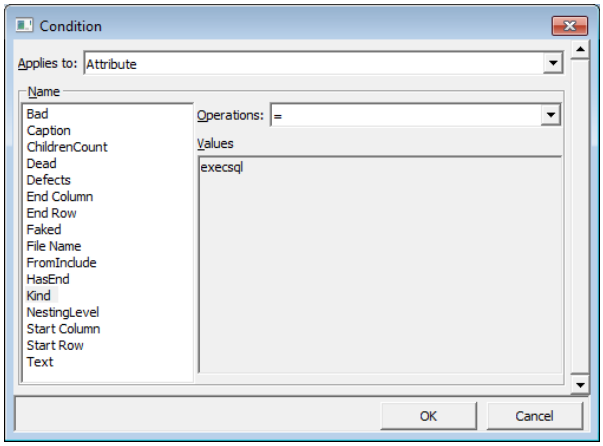
Highlighting the construct type gives you a written explanation of its meaning in the right pane.

6. Select **EXEC Statement** to search for all EXEC SQL statements. These are a subset of all EXEC statements.
7. Press **OK**.



This is the first stage of building your query. Next you will add a constraint because you do not want all EXEC blocks, only EXEC SQL blocks.

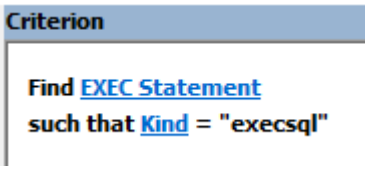
8. Click on the blue **All** to qualify the kind of Exec Statements. You will see the **Condition** dialog box in which you can refine the query.



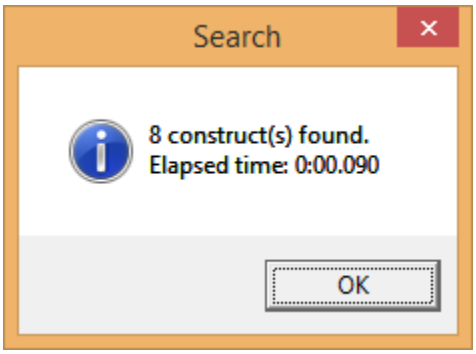
9. Select **Attribute** from the **Applies to:** drop-down box.
10. Select **Kind** in the **Name** box. Kind specifies a subset of the major attribute of a repository entry.
11. Select **=** from the **Operations** drop-down box.
12. In the Values text box, type ***EXECSQL***. This field is not case sensitive.

NOTE: The Condition dialog box is context-sensitive. You might see other options depending on which repository entity you select.

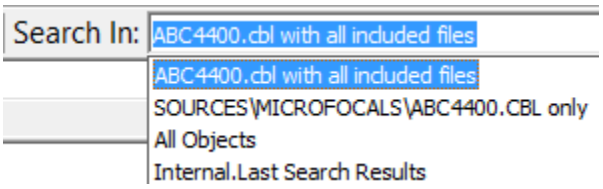
13. Click **OK** to review the query. This will find all EXEC Statements of the kind 'EXECSQL'. 'Kind' is an internal designation by CA, which is why it does not have the space between EXEC and SQL as it does in the actual code.



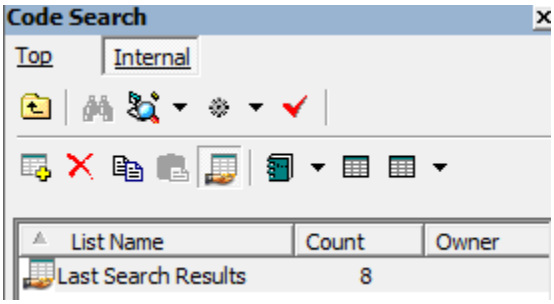
14. Click on **Find All Constructs**.

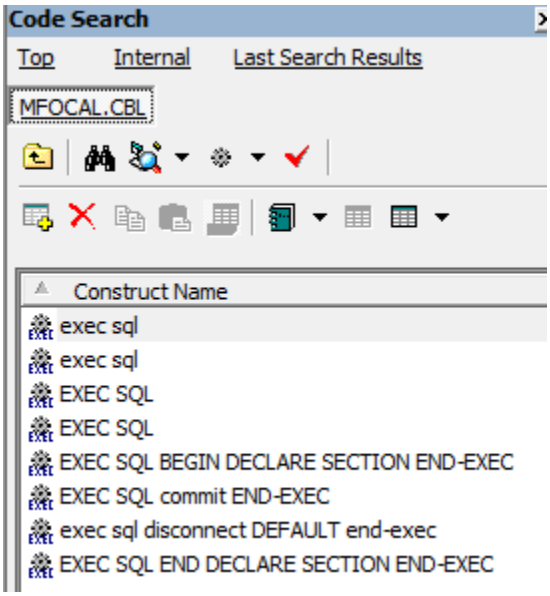


*NOTE: This search is Program-wide. If you want to search Project-wide, you need to change the **Search In** scope or use the custom report feature.*



15. Click **OK** after you have confirmed the search results. You will see the List Browser in Code search with the results displayed for MFOCAL.CBL within the Internal category under the Last Search Results list.

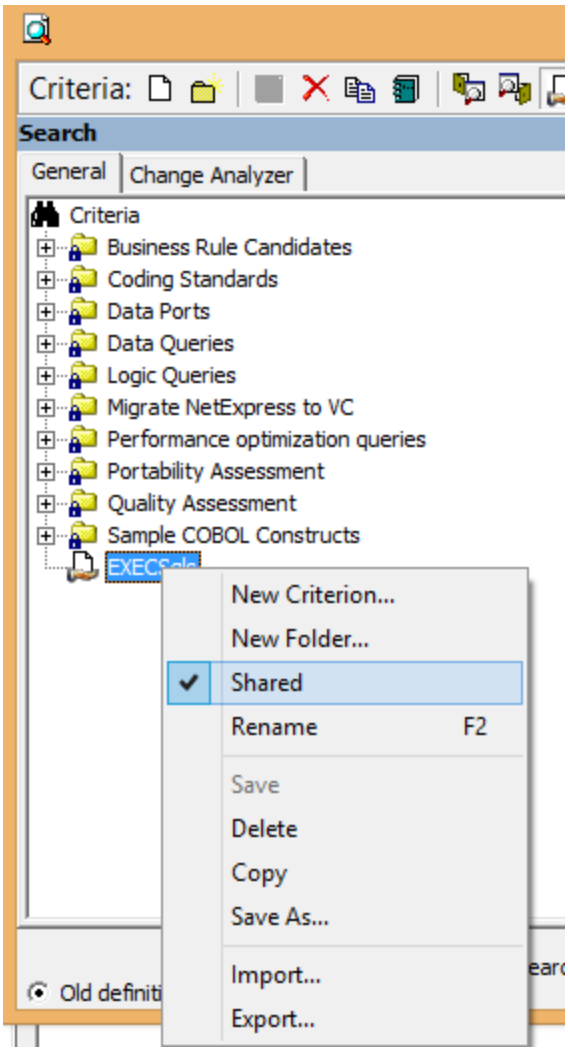




This will always be where the results are placed for program level searches. Observe the list of results. You can re-sort this list by clicking on the column headings ('Construct Name', 'Type' or 'Ln:Col'). You can save, manipulate and retrieve these list contents using the menus within this window.

16. Click the find button again and lcoate your added code search

17. Right click and toggle the shared option

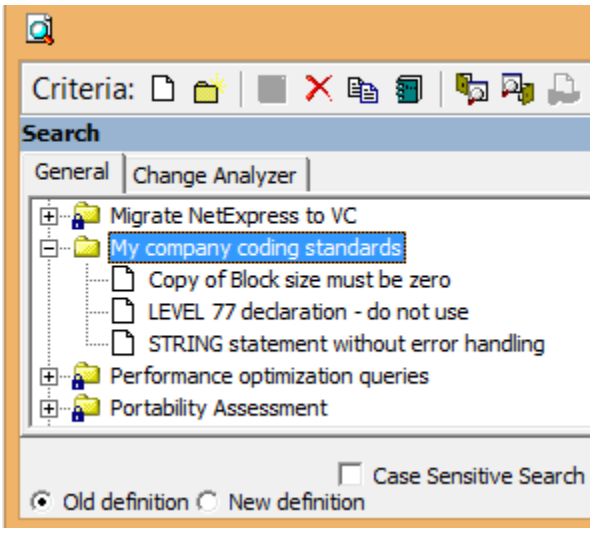


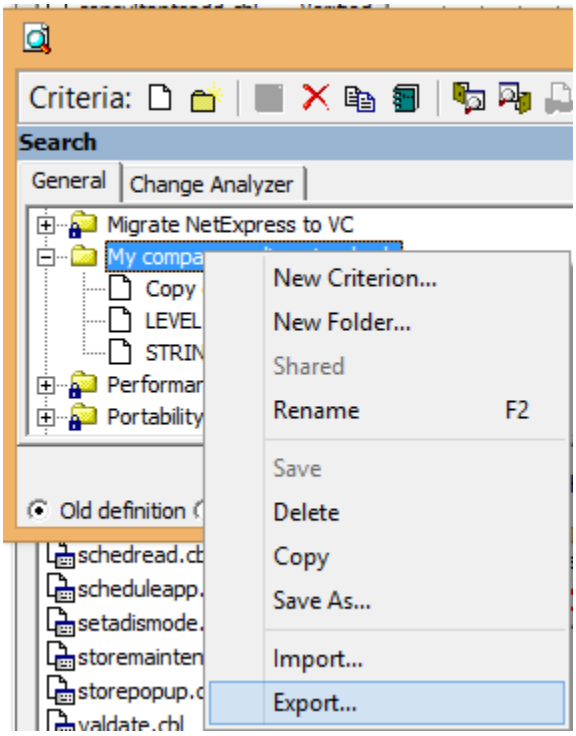
In a central repository setup, shared code searches can be used by anyone who is connecting to the repository.
This is very useful to create a set of code searches to be used by a development team (for example the team's coding standards)

Exporting code searches and folders into Visual COBOL

You can create copies, change and adjust existing code searches to match your needs. These code searches can be exported using the context menu into XML files which can be imported as code analysis rules and rulesets into Visual COBOL for Visual Studio and Visual COBOL for Eclipse

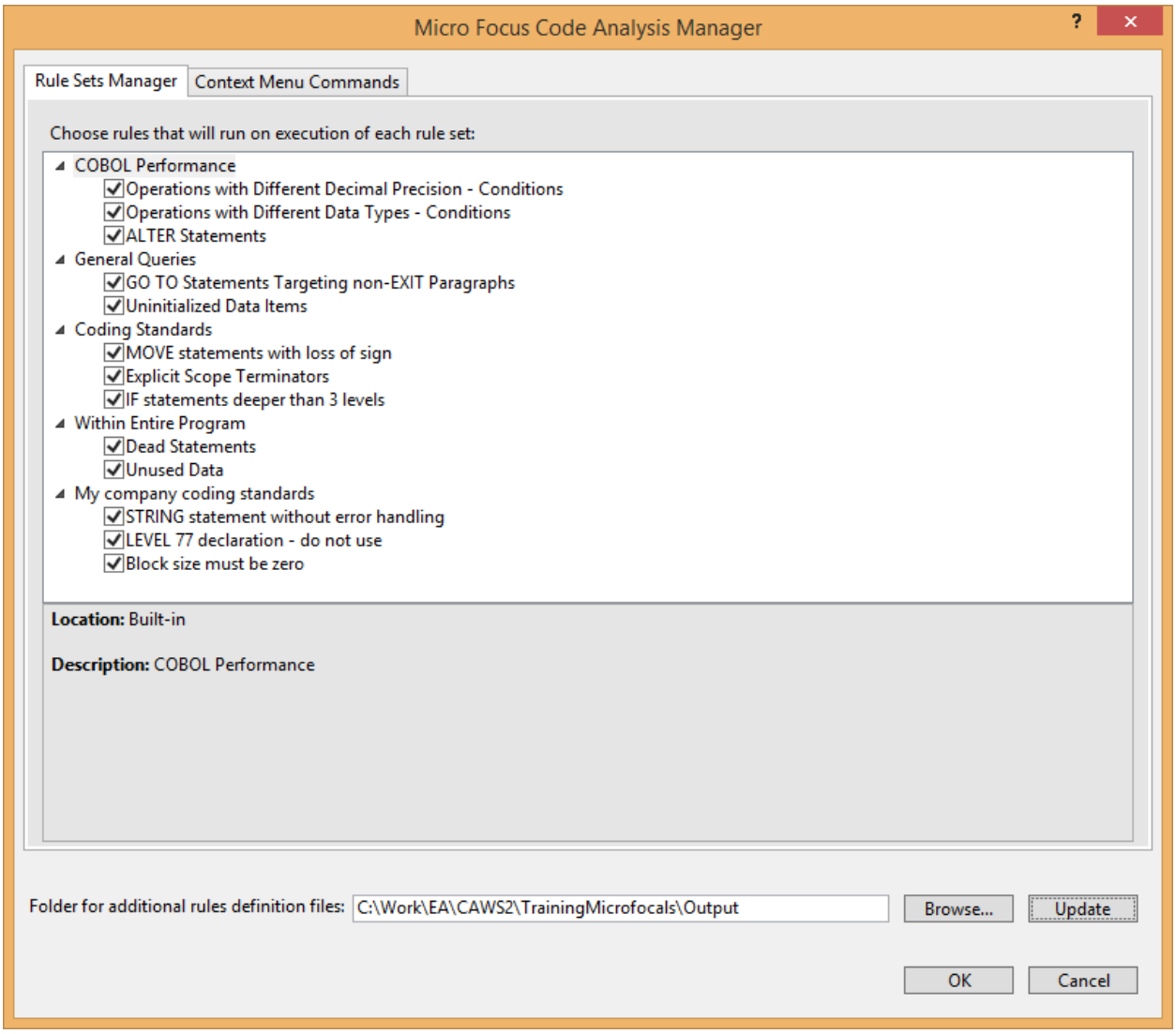
When you arrange code searches in a folder and export the folder, importing this into Visual COBOL will create a ruleset with the same name as the created folder that contains the code searches as rules:



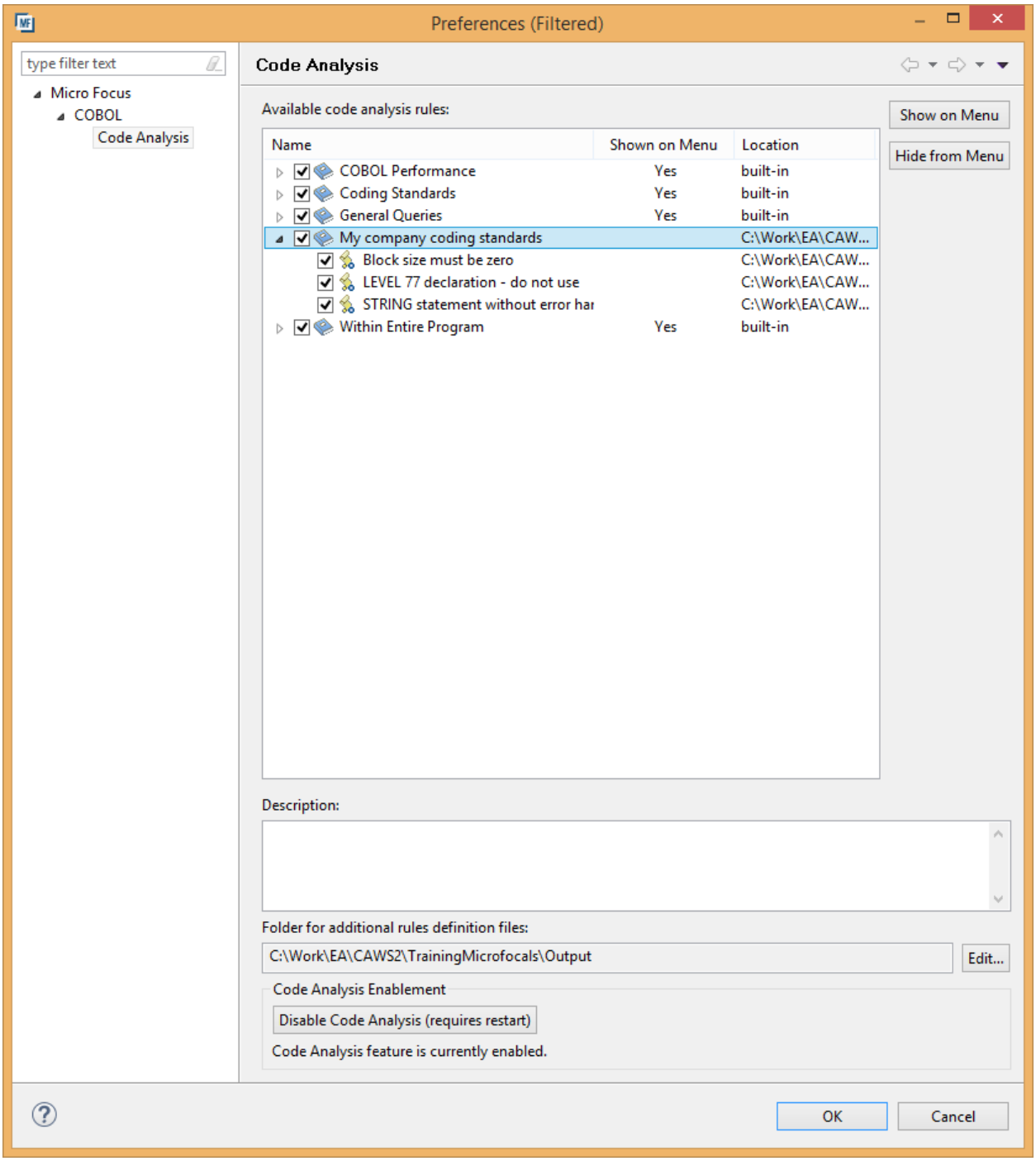


For instructions on how to import into the different IDEs, search for “Importing Rule Sets into the IDE” in the Visual COBOL documentation

The imported ruleset in Visual Studio:



The imported ruleset in Eclipse:

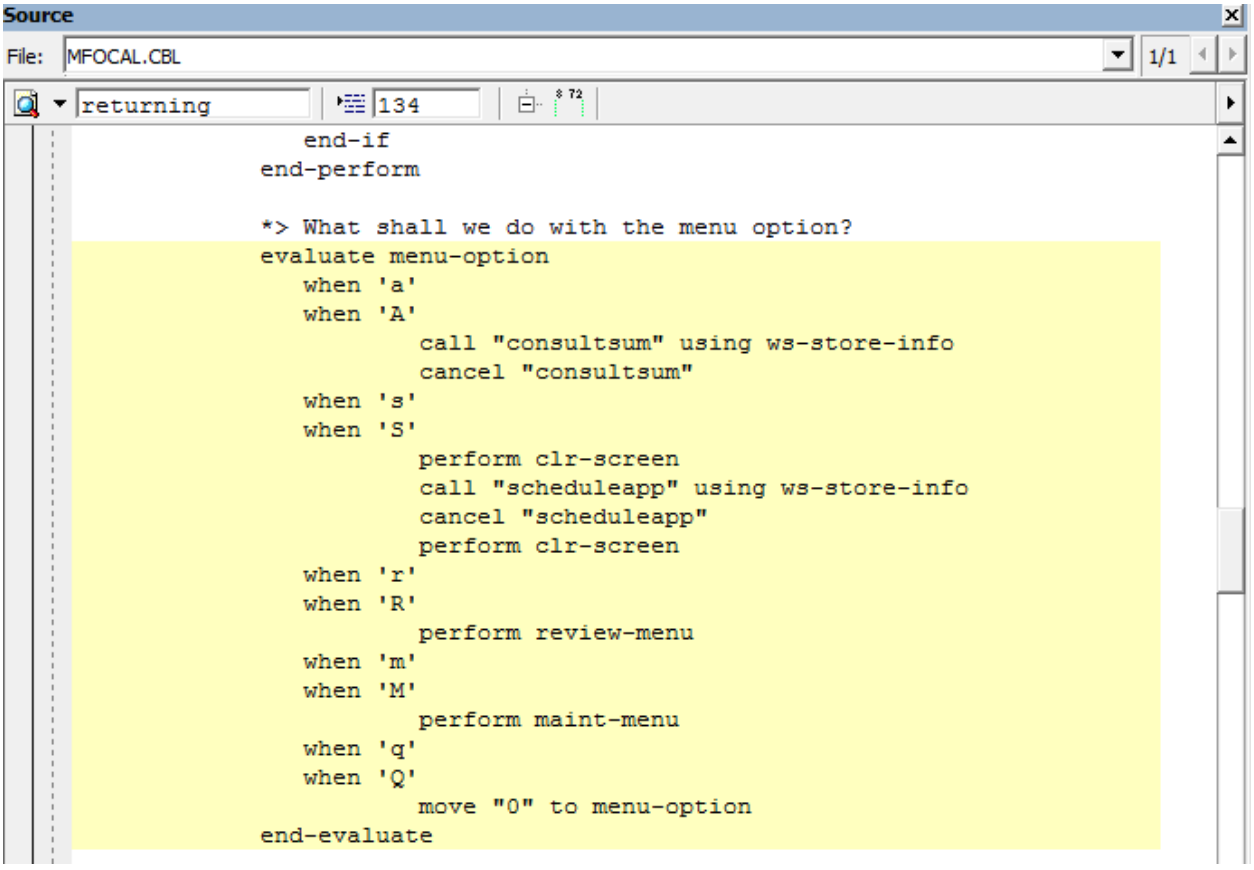


Exercise 7-14: Generating a Query Based on a Similar Query using Code Search

In addition to copying and modifying an existing query, and coding a query from scratch, there is a third way to build a query definition. That is to use an existing coding construct (one that already contains what you are looking for) and then having CA generate a similar query that you can modify and execute.

Objective: You will use Code Search to generate a query to find all occurrences of the checks for a value "a" or "A". Per a fictional Change Request, you have been tasked with changing a related UI option or localization problem.

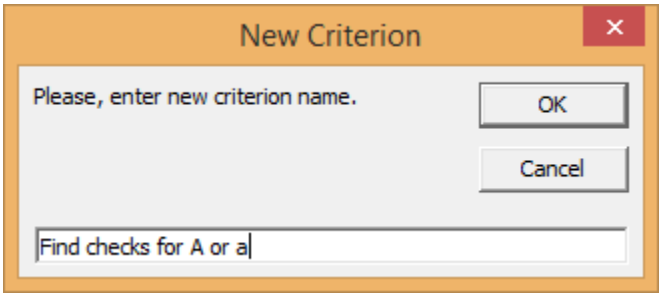
1. To use this feature, you must know at least one area where the code is used. In the Objects pane, select the program **mfocal.cbl**.
2. Navigate to the **line 134**
3. In the Source pane, place your cursor on the **evaluate**. It MUST be on the word **evaluate** and not anywhere else on the line in order to generate the desired query.



4. Make sure you have the **Code Search** pane opened, if not, open it with **View > Code Search**
5. Right-click and choose **Code Search > Search for Similar…** You will see the **New Criterion** dialog box.

Code Search will generate a query and name it based on the coding construct and source member selected.

6. Give the query a meaningful name and press **OK**.

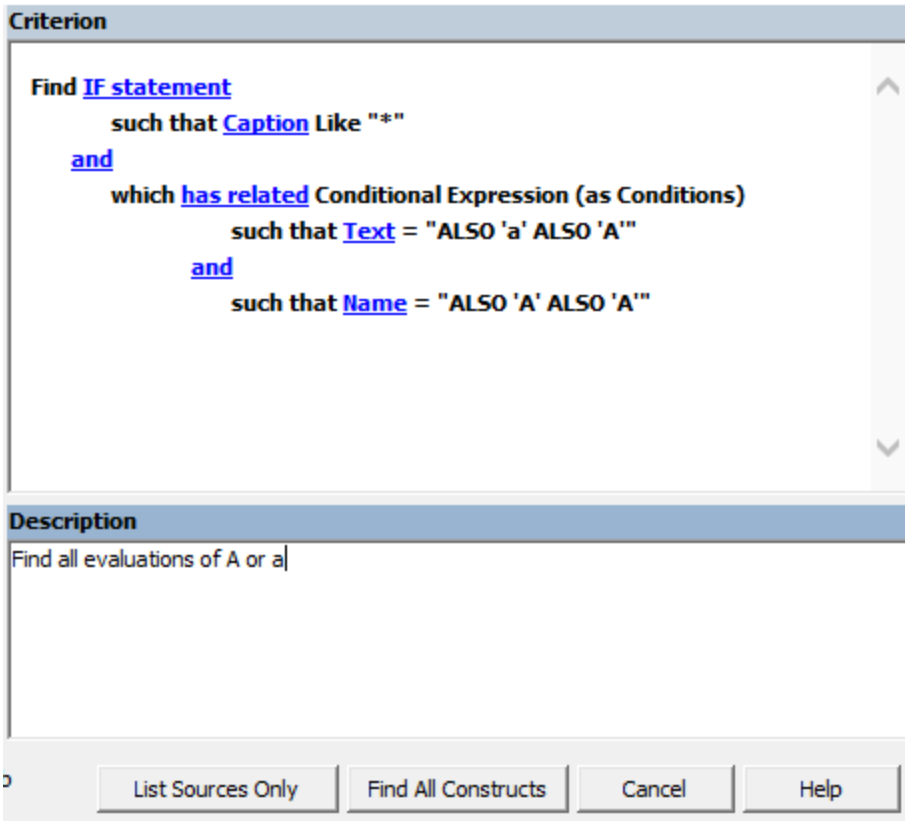


This will generate a query that contains all of the possible criteria for that statement. You might have to delete some criteria that are unnecessary:

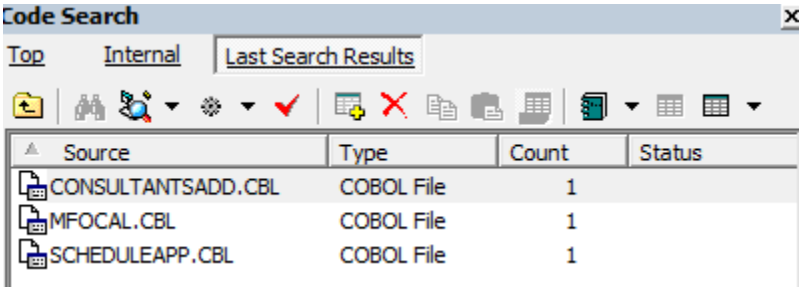
```
Find IF statement
        such that Caption Like "evaluate menu-*"
    and
        which has related Conditional Expression (as Conditions)
                such that Text = "ALSO 'a' ALSO 'A'"
            and
                such that Name = "ALSO 'A' ALSO 'A'"
    and
        which has related Conditional Expression (as Conditions)
                such that Text = "ALSO 's' ALSO 'S'"
            and
                such that Name = "ALSO 'S' ALSO 'S'"
    and
        which has related Conditional Expression (as Conditions)
                such that Text = "ALSO 'r' ALSO 'R'"
            and
                such that Name = "ALSO 'R' ALSO 'R'"
    and
        which has related Conditional Expression (as Conditions)
                such that Text = "ALSO 'm' ALSO 'M'"
            and
                such that Name = "ALSO 'M' ALSO 'M'"
    and
        which has related Conditional Expression (as Conditions)
                such that Text = "ALSO 'q' ALSO 'Q'"
            and
                such that Name = "ALSO 'Q' ALSO 'Q'"
```

7. Click the has related link for **everything but the first one** and choose delete
8. Click on the **Caption** link and choose edit
9. Change the value to *

10. Add a description:

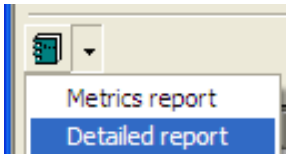


11. Make sure the **Search In** scope is set to **All Objects**
12. Click on find all constructs:



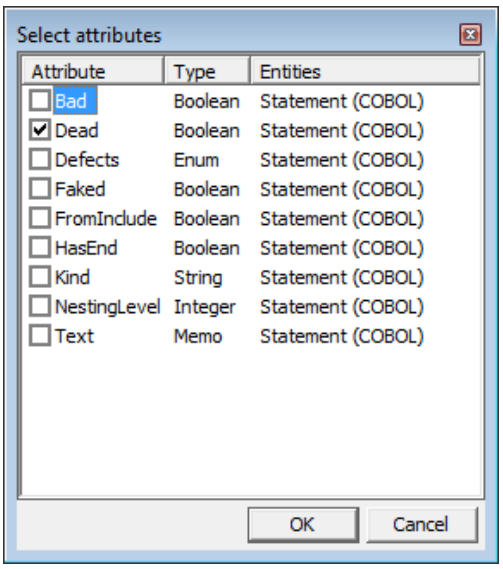
Notice that a simple string search for ***when "a"*** would not have found all of those results because some of the code uses "A" and some uses 'A'. The code search facility is searching the logic and not the syntax in this case.

13. Click on the drop-down arrow next to the Report button and select **Detailed report**.

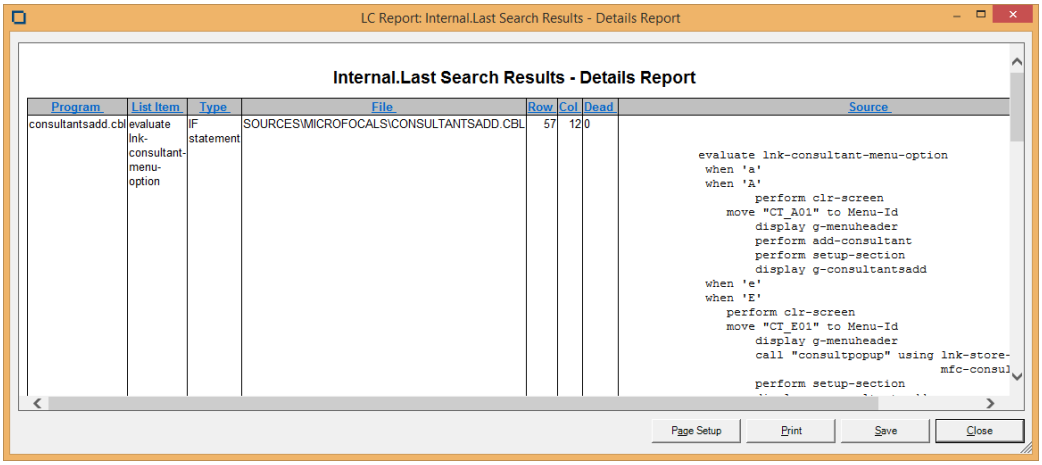


The next step in this scenario would be to provide detailed specifications to the manager or programmer who will be making changes.

14. Check the **Dead** attribute so that it will be included on the report. It is always helpful to know if a line of code you are analyzing is NOT executable.



15. Press **OK**. You should see a report similar to the following.



16. Save the report in HTML format, which typically looks best for this kind of document.

NOTE: Running Queries that use complex relationships such as 'Contains' can impact other users if the Repository is shared, and will produce a warning.

Using the Glossary to Review Conventions

The Glossary provides an organized dictionary of names your application uses to identify data elements. It is useful for assigning natural language business names to identifiers. Glossary lets you assign business names manually or in batch mode. You can also import business names into a glossary from a file.

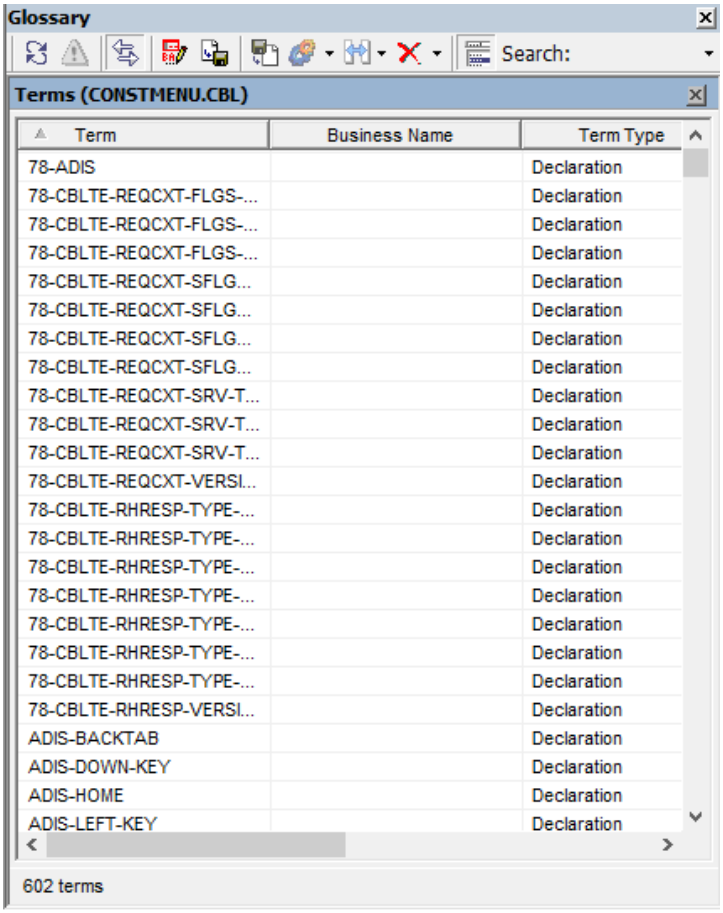
The Glossary provides a good way to review basic naming conventions within a legacy system and to understand which data items are associated with a certain record or file based on their name.

Exercise 7-15: Using the Glossary

This exercise builds on the previous exercise. The intention is to help explain the relationship between tokens and names, as well as the system data naming terminology.

Objective: You will explore the Glossary

1. Select **View > Glossary**. This will open the Glossary.
2. Select **View > Code Search**. This will close the Code Search pane.

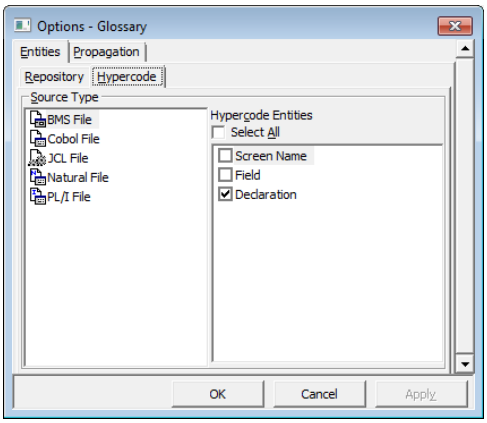


The Glossary tool consists of two panes:

- **Terms Pane** – displays the terms in the selected file
- **Search Result Pane** – displays the terms returned by the most recent Glossary search. The scope of the search is workspace-wide.

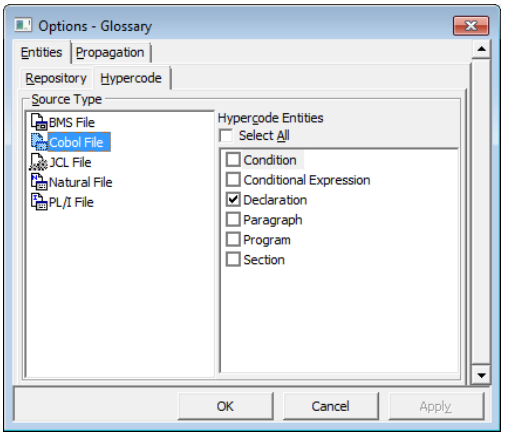
Each pane displays the term type, the business name and description assigned to the term, and how the business name and description was assigned.

3. To hide a pane, click on the **X** in the upper right hand corner of the pane.
4. You can reopen a pane by choosing **View Terms** or **View Search Result** in the Glossary menu.
5. Select **Glossary > Workspace Options…**
6. Click **Entities** and then select **Hypercode**. You will see the Options – Glossary dialog box.



The Options window allows you to select what you wish to display in the Glossary, either at the Repository or Hypercode model. The Hypercode Tab varies depending on the languages selected in CA configuration.

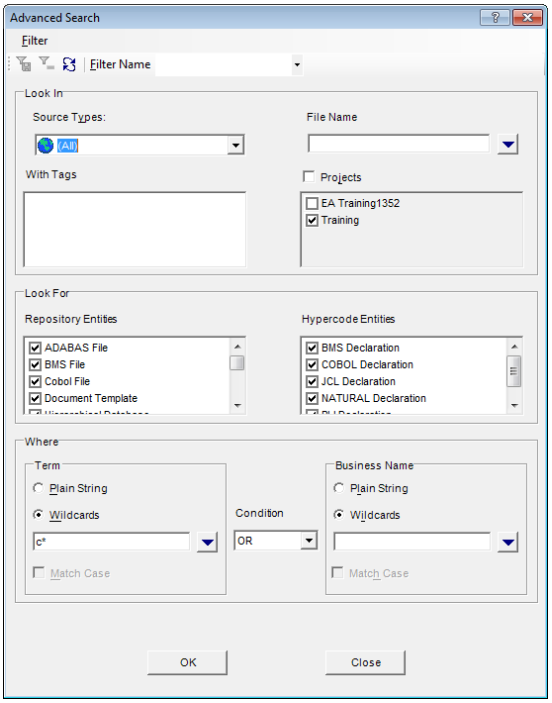
7. Select **Cobol File** as the Source Type.



This specific window allows you to choose which identifiers you wish to see in Glossary. By default, only the Declarations are enabled for each of the various file types.

8. Click **OK** to close the window.

9. Select **Glossary > Advanced Search**. You will see the **Advanced Search** dialog box.



The Advanced Search feature can be used to only display terms from certain files. This is designed to further refine what is shown from the **Options** window mentioned above. These options allow you to filter terms based on Source Type, Source Name, Repository Object, Hypercode Object, or Entity Name. It is also possible to filter by Project name or objects associated with certain tags.

For this exercise, leave the options set to their defaults.

10. Click **Close** to close the **Advanced Search** dialog box.

Adding Business Names to Data Items

In some cases, managers and casual users may find it difficult to understand the meaning behind the name of a data item. CA allows you to associate “business names” with a data item, which is essentially a description of the field. To truly be effective, business names must be applied to *every* data element name that is not self-explanatory.

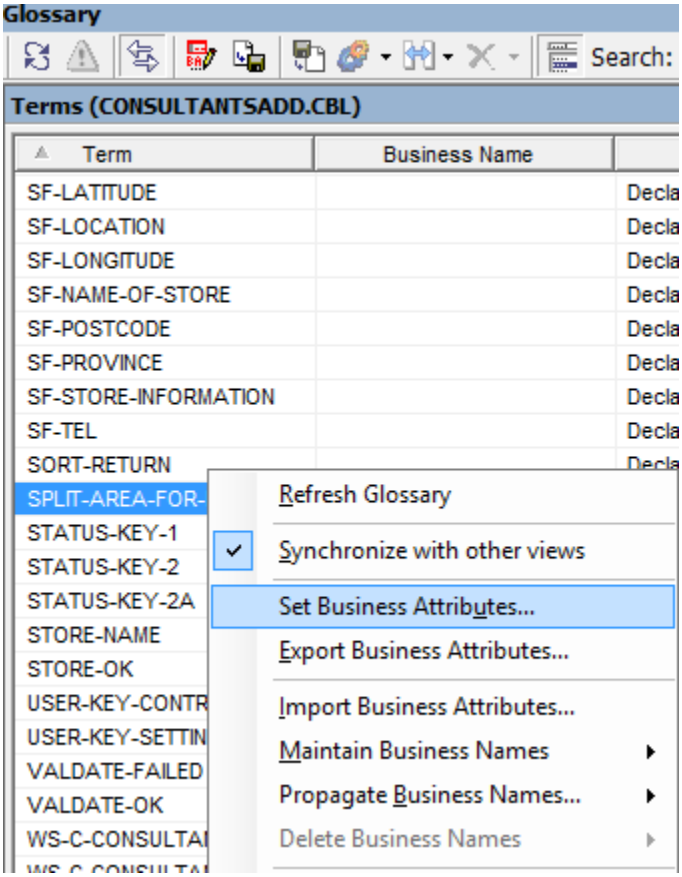
Exercise 7-16: Associating Business Names with Data

In the next exercise, we apply some Business Names manually to data items and generate a report to view them. We will also explore assigning business names in batch mode. Adding business names to programs, paragraphs and data names can have a significant effect on the understanding of the system. It is highly recommended that this feature is used (select “Display Business Names” in User Preferences.

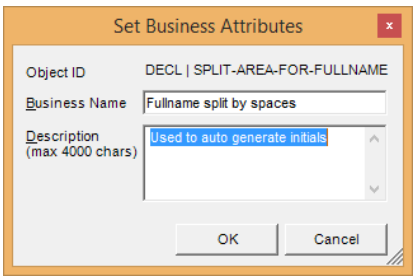
NOTE: Master Users of CA have the ability to automatically propagate Business Names at the Token level which affects all Terms. Standard Users are restricted to manually applying Business Names against Terms.

Objective: You will apply Business Names to data prefixed with CM.

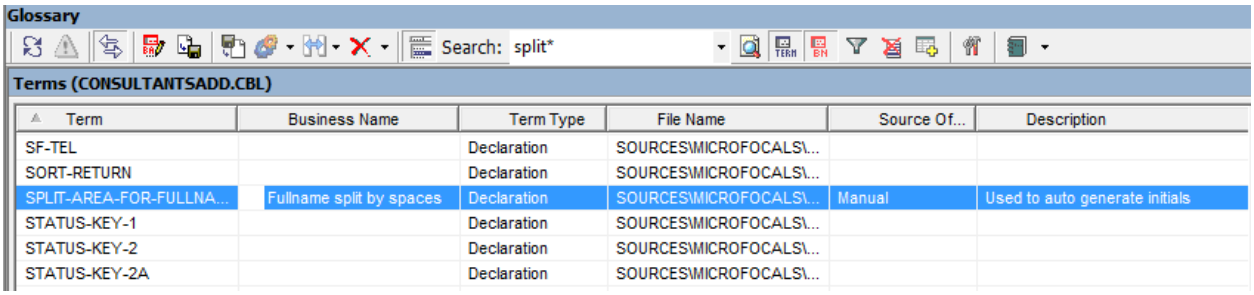
1. In the objects pane, switch to **consultantsadd.cbl.**
2. In the Terms pane, use the scroll bar to move down to **SPLIT-AREA-FORFULLNAME**.
3. Right-click on **SPLIT-AREA-FORFULLNAME** and select **Set Business Attributes**.



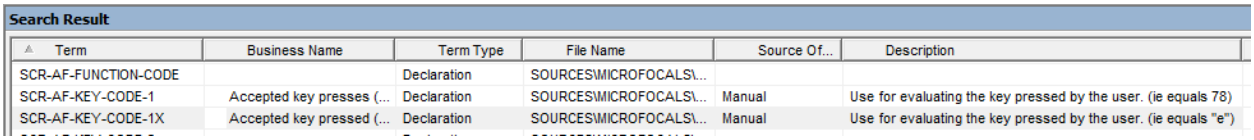
4. Type in a descriptive Business Name, such as **Full name split by spaces**.
5. Type in a meaningful Description, such as **Used to auto generate initials**.



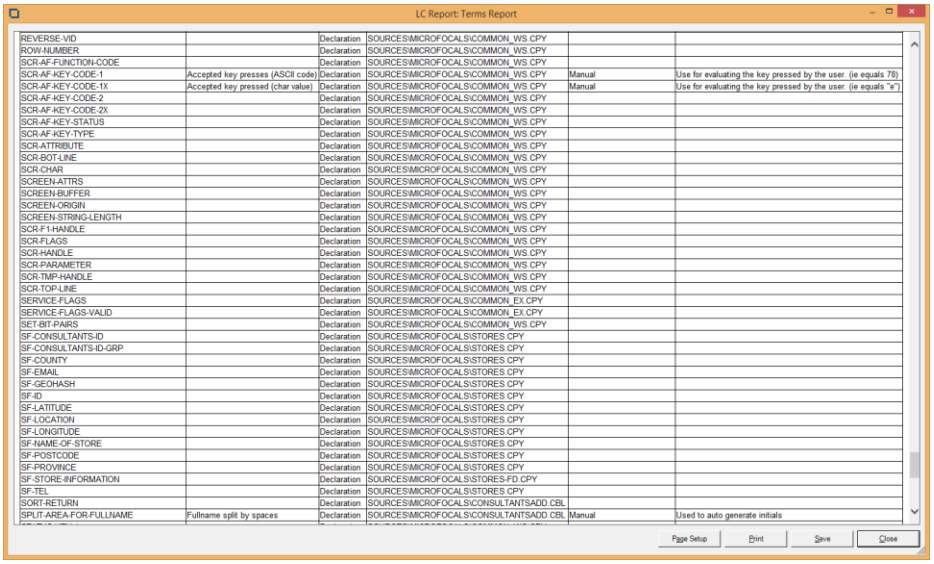
6. Press **OK**.
7. Adjust the column headers in the **Names** pane by sliding the bars to the left. You may also have to resize the **Names** pane to view the complete Business Name. This allows you to view the business name that was just applied.



8. Make sure the TERM and BM (Business name) toggle buttons are toggled and search for scr* and add some descriptive information for scr-af-key variables:



9. Select **Glossary > Terms Report** and scroll to find the information we've added:



Business names can also be added for Paragraph and Section names and also for Programs. To do this you need to open the **Options** to include these in the Glossary Terms listed. There is a benefit to doing this as these business names can be shown on diagrams and also appear in the Context view.

10. When you are finished viewing the report, click **Close**.

11. Business names can also be automatically propagated when COBOL Analyzer discovers assignments that are used in the code. For example, add a business name and description to the term LNK-NAME-OF-STORE.
12. Right click on the term and choose: Propagate Business Name→Propagate Selected
13. Notice that the business name and description have propagated to STORE-NAME as well.

NOTE: You can add information to the program glossary and then select Propagate Business Name→Propagate All to let COBOL Analyzer fill additional information based on yours.

14. Close Hyperview by clicking the **X** at the top right-hand corner of the window.

Change Analyzer

CA allows you to identify sources that may be impacted by modification of a data item's internal definition or its usage. Examples of changes of this type would include expansion of a database field or report field. In order to understand which data elements a change like this would affect, you will need to know which data elements are affected in the database as well as intermediate fields that may contain or use the data element in calculations.

Using an advanced search criteria, you can search for data items based around partial names and patterns with wildcard support. Once you have found your data items, you can categorize the results into lists where further impact analysis can be performed.

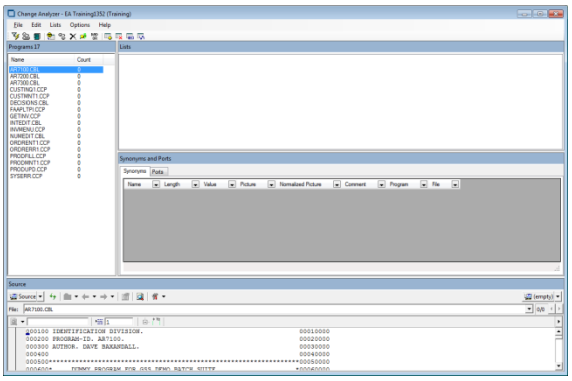
In this section, you will examine data items that are potentially impacted by some type of change to your application system.

Exercise 7-17: Using the Change Analyzer

Suppose you are an analyst programmer who is thinking of making changes to the structure of the Customer Master record. You would want to see which data items you would have to change, as well as other data elements that may be affected (and in turn, require a change).

Objective: You will create a search for time-slot related items in a project and locate related data items for potential impact analysis that could result from a change.

1. Select **Analyze > Change Analyzer**.



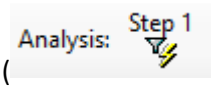
The Change Analyzer tool consists of 4 views or panes:

- **Programs**
- **Lists**
- **Impact Tracing**
- **Source**

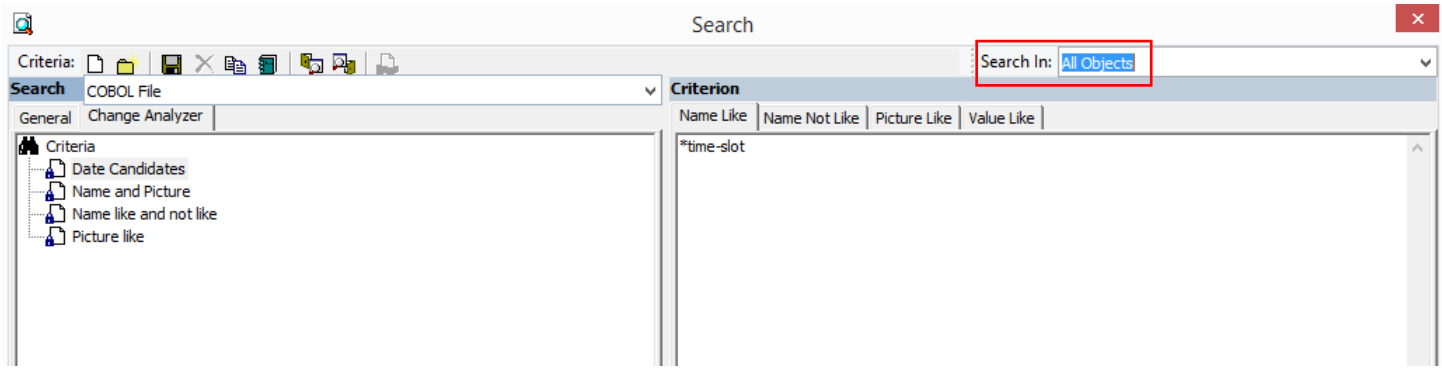
The **Programs** Pane is populated when Change Analyzer is opened. It contains an entry for each program in the project that is parsed to enable analysis with Change Analyzer. Selecting any program in this pane will alter the contents displayed in the **Source** pane and may alter the contents displayed in the **Lists** Pane.

Programs 26

Name	Count
ABC4400.CBL	0
AUTOSETUP.CBL	0
CENTERTEXT.CBL	0
CONSTMENU.CBL	0
CONSULTANTSADD.CBL	0
CONSULTPOPUP.CBL	0
CONSULTSUM.CBL	1
CUSTMAINT.CBL	0
CUSTMENU.CBL	0
CUSTPOPUP.CBL	0
DATEINFO.CBL	0
ERRPOPUP.CBL	0
GENCUSTID.CBL	0
GEOHASHENC.CBL	0
GETCONSULTS.CBL	0
GETDEFSTORE.CBL	1
GETONESTORE.CBL	0
GETTIMESLOT.CBL	1
MFOCAL.CBL	1
MFOCALHELP.CBL	0
SCHEDREAD.CBL	1
SCHEDULEAPP.CBL	1
SETADISMODE.CBL	0
STOREMAINTENANCE.C...	0
STOREPOPUP.CBL	0
VALDATE.CBL	0

2. Press Step 1 in the toolbar (

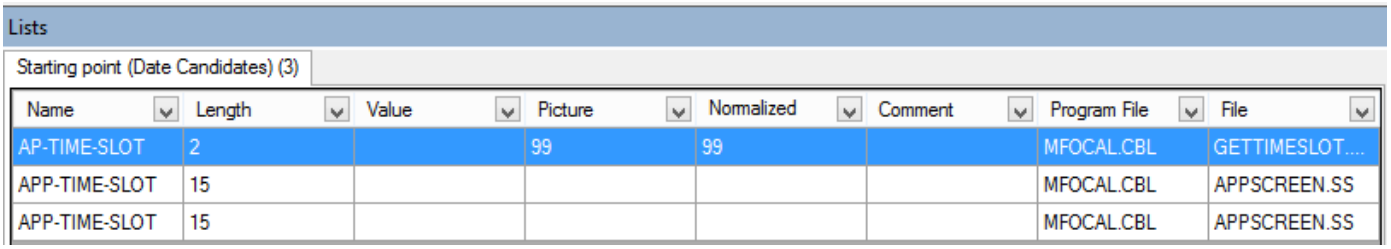
) and click OK on the message box that will appear
3. A search pane will open where you can use the code search capabilities to build a list of possible items to be included in the starting point list.
4. Choose the data candidate and change the Name Like criterion to ***time-slot.**



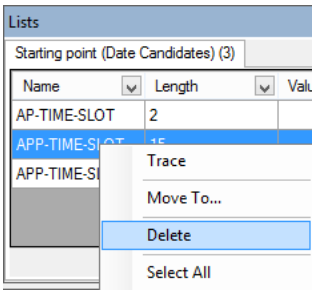
5. Make sure the **Search In** scope is **All Objects**. Click **Find All Construct**

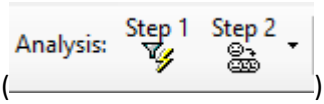
Tip: If you have a specific variable declaration that you are going to change, you can search for it in the repository browser and invoke Change Analysis for this item from the context menu as well.

6. The programs pane will contain a count of all the found declarations, choose the different programs to review the results.
7. Go to mfocal.cbl

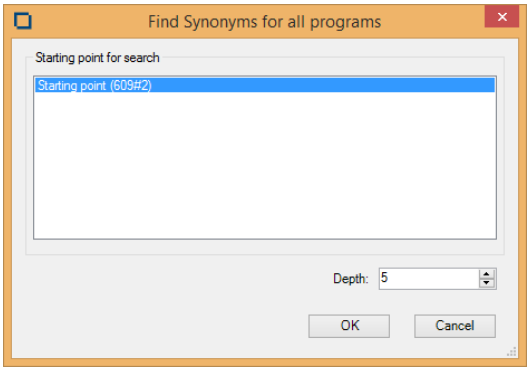


8. The list contains 3 items, let's decide that APP-TIME-SLOT are not relevant and we're not going to make any changes to those declarations. Right-Click on them and Delete:



9. Now, we have all the starting points reviewed and we're happy that we're going to make changes to all of them, Click Step 2 in the toolbar (

)
10. You can select to do the analysis for all the programs or just the selected one. In this example, we'll analyze for all the programs.



Tip: Changing the Depth of analysis impacts accuracy and performance. Depth will be explained in step 13.

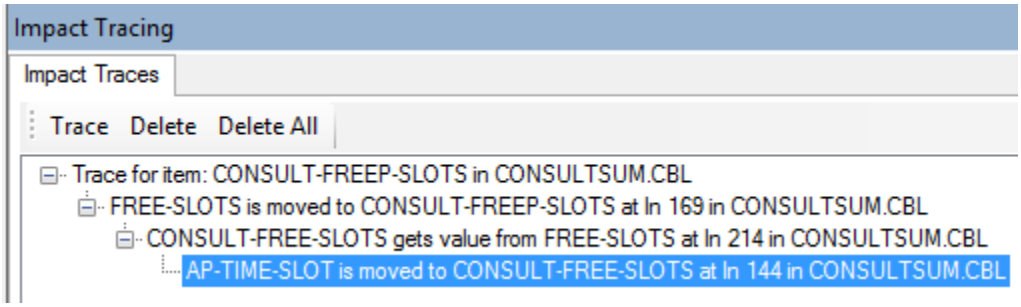
11. Go to CONSULTSUM.CBL and review the affected by list:

Lists

Starting point (Date Candidates) (1) | Affected by AP-TIME-SLOT (GETTIMESLOT) (9)

Name	Length	Value	Picture	Normalized	Comment	Program File	File
AP-TIME-SLOT	2		99	99		CONSULTSUM....	GETTIMESLOT....
AP-CURRENT-TI...	15					CONSULTSUM....	GETTIMESLOT....
CONSULT-FREE...	8		99	99		CONSULTSUM....	CONSULTSUM....
FREE-SLOTS	8		99	99		CONSULTSUM....	CONSULTSUM....
COUNTER	4		9(9)	9(9)		CONSULTSUM....	CONSULTSUM....
MAX-APPS-PER-...	2	18	9(2)	99		CONSULTSUM....	COMMON_78.CPY
CONSULT-FREE...	8		999.99	999.99		CONSULTSUM....	CONSULTSUM....
BUSY-SLOTS	8		99	99		CONSULTSUM....	CONSULTSUM....
CONSULT-BUSY...	8		99	99		CONSULTSUM....	CONSULTSUM....

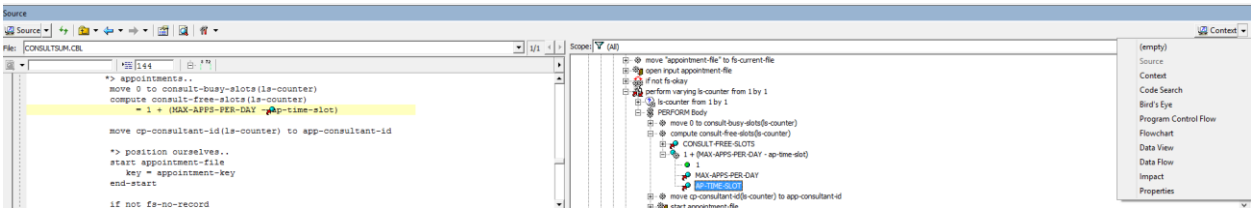
12. When you don't understand why some data item is affected by the changing AP-TIME-SLOT, you choose it and click the Trace button in the Impact Tracing pane.
13. Select CONSULT-FREEP-SLOTS, click the trace button and expand the added tree:



This tracing explains how changing AP-TIME-SLOT might require a change in CONSULT-FREEP-SLOTS as values are indirectly used to affect its value.
This trace for example **is of depth 3**, which means the tool traced 3 level of data assignments to reach the decision that CONSULT-FREEP-SLOTS is affected by a change to AP-TIME-SLOT.
Raising the Depth in step 10 will reach more affected code, but will take longer to analyze, especially when there is a large number of starting points.

14. Clicking on each entry in the trace will navigate to the matching place in the code, and clicking on each item in any list will navigate to its declaration.

Notice that you can add another pane in the source pane, to help you better understand the code or do further analysis before you decide if you want to keep results in the list or remove them:



15. Remove BUSY-SLOTS from the list by right-clicking and delete.

Exercise 7-18: Document a System-Wide Data Element Change

Suppose that now, as an analyst/programmer, you are required by a manager to produce a summary of your impact analysis review in some type of deliverable format, or you just wish to have a detailed summary of the results of your review to aid in the planning and preparation for a system change.

At this stage, CA will produce reports that give an overview/summary of the Impact Analysis review, which is useful for reporting to a project manager or planning the next phase in making a system wide change to your data items.

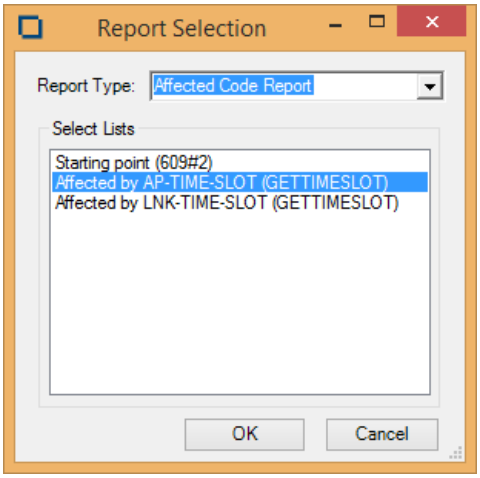
Objective: You will generate an Affected Code report.

1. Select **File > Reports**. Or you can click the **Report** button  .

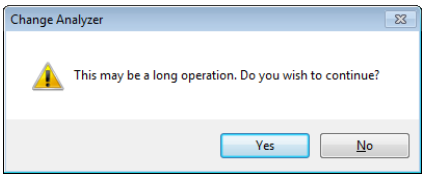
   There are a number of reports available in Change Analyzer:

   - **Report on Selected Program** provides a report based on the selected program in the Programs pane for the list selected
   - **Report on The Whole List** provides a report on the selected list for all programs in the project
   - **Metrics Report** provides counts of the number of data items in each list for all of the programs in the project
   - **Affected Code Report** provides a report of the affected lines in every source file highlighted in bold text with a number of lines before and after for context – see Project Options on how to determine the neighborhood size.

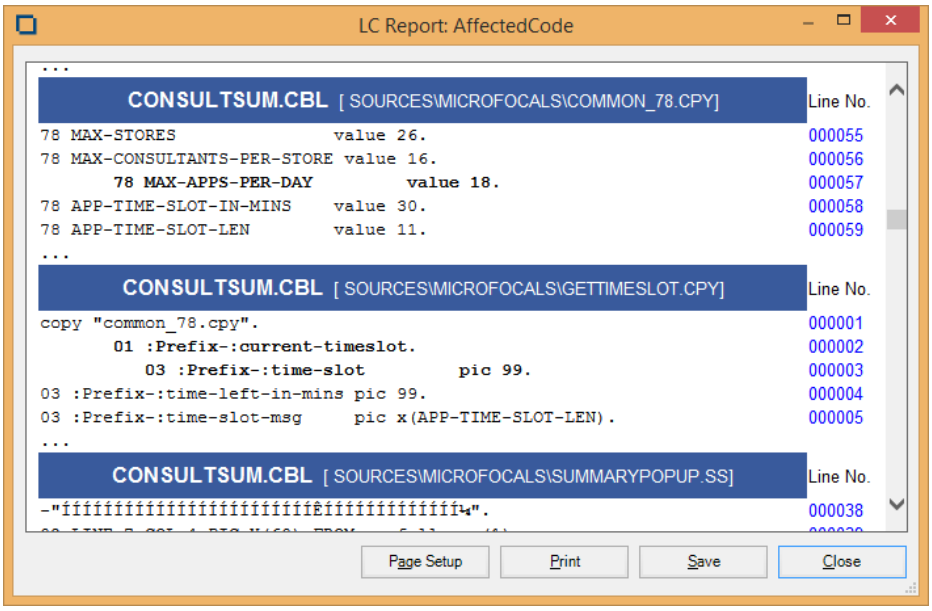
2. Select the **Affected Code Report** and base the report on the items in the **Affected** list.



3. Click **OK**. You will see the **Change Analyzer** dialog box asking if you want to continue.



4. Click **Yes**.



5. Click **Close** to close the report.
6. Close the Change Analyzer tool.

Summary

In this module, you learned:

- How to use the Diagrammer tool
- How to define relationships you want to diagram
- How to use the Interactive Analysis (Hyperview) tool
- How to use code searches and code search reports
- How to use the Glossary to review conventions
- How to add Business names to data items
- How to use the Change Analyzer