COBOL-IT Developer Studio

Release notes

Items marked with (E) are only available to users with Subscriptions to the Enterprise edition.

Please note:

Upgrading to a Major Release (1.0, 2.0, 3.0, etc…) requires recompilation of all programs.

Minor Releases (2.x, 3.x, etc…) document recently added features in maintenance releases (2.x.x, 3.x.x, etc…). Note that in exceptional cases, a minor release may contain updates that require recompilation of all programs. Where this is the case, this will be prominently featured in the Release Notes.

2.0

2.0.4

New

2.0.4 COBOL-IT Web Services

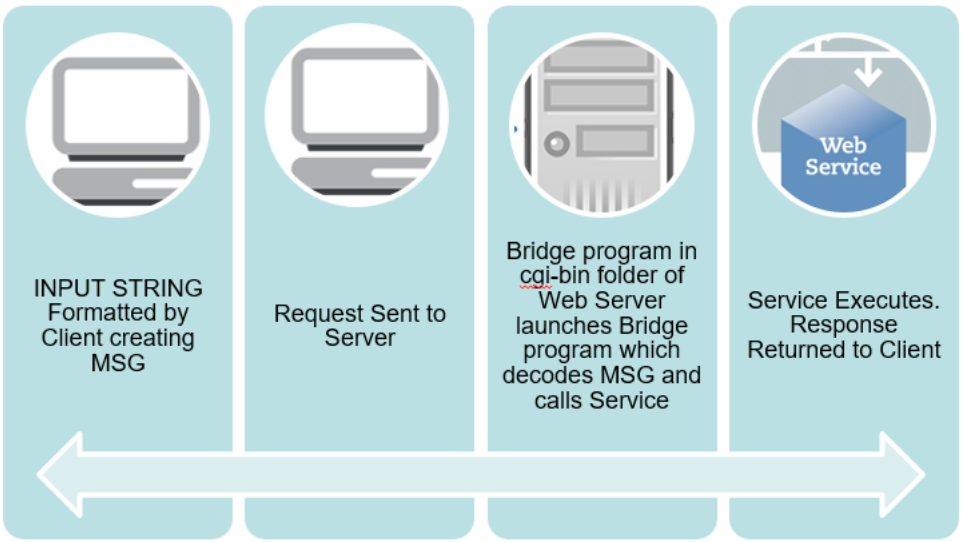
The COBOL-IT Web Services Perspective in the Developer Studio provides a platform that allows the user to convert a COBOL-IT Web Service program into a callable Web Service. The RESTful Web Service functions, POST, GET, PUT, and DELETE, map well to common COBOL operations. A single COBOL program can have entry points mapped to each of these functions.
In our samples, we will be using indexed files as the target of the COBOL operations.

RESTful Web Service function	COBOL operation
POST	OPEN [OUTPUT/I-O FILE], WRITE
GET	OPEN [INPUT FILE], READ
PUT	OPEN [I-O FILE], REWRITE
DELETE	OPEN [I-O FILE], DELETE RECORD

The COBOL-IT Web Services Perspective provides two outputs which allow the COBOL-IT Web Service program to be converted into a callable Web Service. These are the INPUT STRING, and the BRIDGE PROGRAM. The COBOL-IT Web Services Perspective supports three different configurations, the Wildfly Application Server Configuration, the Apache Web Server Configuration and the xbind configuration.

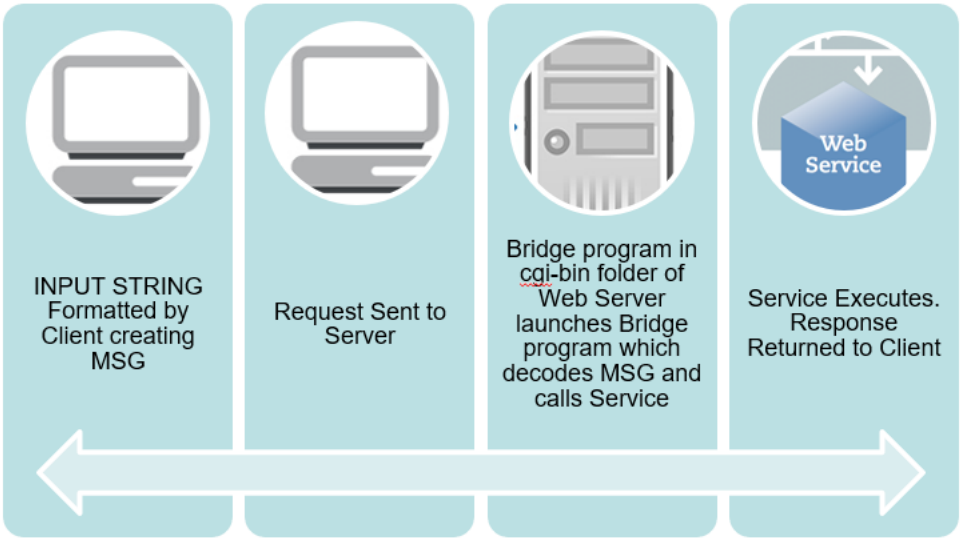
The WildFly (JBoss) Configuration

In the WildFly Application Server Configuration, the Bridge Program is executed from within a script located in the … folder of the Application Server. In our example, the shell is identified as: form action=http://… in the html file executing on the client. The shell script is required because COBOL-IT must be able to locate the license file, and the COBOL-IT environment must be setup.



The Apache Web Server Configuration

In the Apache Web Server Configuration, the Bridge Program is executed from within a script located in the cgi-bin folder of the Web Server. In our example, the shell is identified as:
form action=http://localhost/cgi-bin/xholidays41.sh in the html file executing on the client. The shell script is required because COBOL-IT must be able to locate the license file, and the COBOL-IT environment must be setup.



The xbind Configuration
In the xbind Configuration, the Bridge Program is executed directly by the xbind listener. In our example, the listener is launched from a shell script.

```
./xbind 9735 xholidays &
```

and identified as:
form action="http://localhost:9735" in the html file executing on the client. The shell script is required because COBOL-IT must be able to locate the license file, and the COBOL-IT environment must be setup.

2.0.1 Changes to licensing

COBOL-IT has made changes to licensing to all products, effective with the the release of COBOL-IT Compiler Suite version 4.0 (and later). With the release of COBOL-IT Compiler Suite version 4.0, separate license files are required for each product/platform pairing. As a consequence, users can no longer use the same license for multiple products, on multiple platforms; users deploying multiple products require multiple license files.

Default location
The default location for COBOL-IT product license files in %COBOLITDIR% (Windows) and $DEFAULT_CITDIR (Linux). For COBOLIT Compiler Suite version 4, the default installation directory is /opt/cobol-it4-64 (Linux).

Default naming convention
License files located in the default location named “citlicense.xml”, or with names prefixed by “cit-license-“ and with the “.xml” extension will be validated by COBOL-IT products.

Using COBOLIT_LICENSE to reference single or multiple license files
For cases where different naming conventions are used, or where license files are not stored in the default installation directory, the user should use the COBOLIT_LICENSE environment variable to indicate the full path(es) and name(s) of their license file(s).

Note that when indicating multiple license files, the semicolon “ ; ” separator is used. In Linux, the list of license files is started and finished with single-quote marks “ ‘ “. The single-quote is located on the same key as the double-quote on most keyboards.

As examples (Linux) :
>export COBOLIT_LICENSE=/opt/cobol-it4-64/compilerlic.xml
>export COBOLIT_LICENSE='/opt/cobol-it4-64/compilerlic.xml;/opt/cobol-it4-64/citsqllic.xml'

2.0.0 Refactoring

Developer Studio version 2.0 upgrade of the installed software libraries underlying many of the functionalities of the Developer Studio. In some cases, older versions of libraries were deprecated, in favor of newer versions. Newer libraries generally have better performance, and better support. In all cases, the function of testing was to ensure that prior functionality was preserved.

2.0.0 Version Matrix

Version 2.0 of Developer Studio can be used with Eclipse Kepler or with Eclipse Neon/Java 8.

Features that require Eclipse Neon/Java 8 include :

Developer Studio	Eclipse	Java	Comments
2.0	Kepler	Java 6+	No RSEGit . Includes Code Coverage, Profiler, Data Displayer, support for Git and Mylyn
2.0	Neon	Java 8	Includes Code Coverage Data Displayer, Profiler, support for Git and RSEGit, and Mylyn

2.0.0 Migrated to the latest RSE version

The Developer Studio is now built with the latest RSE version. In addition to increased stability, certain icons that were not available in the previous version are now availalble.

2.0.0 Migrated to latest TM terminals version

The Developer Studio no longer uses the obsolete "terminals ui" component, and is now built using the latest TM terminals version.

2.0.0 Data Displayer

The Data Displayer supports the browsing of relative, line sequential and binary sequential files, and the browsing and editing of indexed sequential files. Users can adapt the display characteristics by selecting from multiple KEYs described in the SELECT statement, from multiple RECORD formats described in the FD, and by setting START keys and conditions.

You can browse your file either in a table mode, in which a pre-selected number of records are displayed, or in a single-record mode, in which a single record is displayed.

The COBOL-IT Data Displayer uses an EXTFH server, and Data Dictionary files (**XDD**s) to interpret and display data files. To generate an XDD, compile a COBOL program that includes the FD for your file, including the –fgen-xdd compiler flag.

The File Records Toolbar of the COBOL-IT Data Displayer provides toolbar buttons that allow you to Open a file, select Next/Previous Record, Configure the Columns, and Close a file.

2.0.0 Code Coverage

Code Coverage records which parts of your COBOL code are executed during a particular program launch. Coverage analysis is very simple:

1. Compile your source files with -code-cover -debugdb debugdb.dbd
2. Run the program

3. Analyze coverage data

Code Coverage records which parts of your COBOL code are executed during a particular program launch. Programs compiled with the code coverage compiler flag produce coverage information that is presented in the Coverage view. Source Code in the COBOL Code Editor is automatically decorated to demonstrate whether lines of source were executed or missed in the run.

2.0.0 Profiler

The Profiler records where your application uses CPU time, elapsed time, memory, and CPU processing power. Gathering Profiling data is very simple:

1. Compile your source files with -fprofiling
2. Run the program
3. Analyze coverage data

Programs compiled with the profiling compiler flag produce profiling data that is presented in the Profiler View which includes Runtime and Paragraph tabs. In the Runtime tab, you have access to Memory and CPU usage, displayed in real time. In the Paragraph tab, you have access to paragraph-oriented statistical data, including number of times entered, time elapsed, CPU time elapsed, external calls, and time elapsed in external calls.

2.0.0 Git Source Code Control

The Git Perspective provides the ability to create a source code repository locally, or to access an existing source code repository hosted on the web. In addition to being the most widely-used open source source code control system, Git provides mechanisms to set triggers for automated test servers such as Jenkins operating in the DevOps delivery cycle.

2.0.0 RSE Git

RSE Git functionality requires an RSE connection and an existing Git repository on the remote machine. With Developer Studio 2.0.0, users can create a remote project using the remote Git repository, and have access to a broad range of Git functionalities such as fetch, pull, push, checkout, switch to, and commit functions.

2.0.0 Mylyn Task Manager

Mylyn is the task and application lifecycle management (ALM) framework for Eclipse.
With connectors to popular issue management tools, Mylyn integrates other ALM tools into the Developer Studio as well, allowing it to greatly enrich the developer's experience.

2.0.0 Profiler>Support for Profiler in Dev Studio / Kepler Build

The Profiler Paragraphs tab is available in the Kepler build. The Runtime tab of the Profiler View is not supported in the Kepler Build.

2.0.0 Profiler>Link to Source Code function

The Link to Source Code function in the Profiler allows you to return to the line of source being executed when the function is clicked during the Profiling of an application.

2.0.0 Profiler>Columns in the Profiler View added

For both CPU and Elapsed Times, there is a new "Intrinsic Avg Run" column, which displays a computed time.

2.0.0 Profiler>Profiler and Debugger together

You can use the Profiler and Debugger together.

2.0.0 Profiler>Export Profiler file function
The Export function can now be applied to the Profiler, with the result being that the CSV file can be exported to a chosen location.

2.0.0 Git>Replace with File in Git Index
Changes which are not yet committed and not yet staged can be reverted for a set of selected files. Select the file(s) in the Navigator or an analogous view and click **Replace With > File in Git Index**.

2.0.0 Git>Replace with Previous Revision
Changes that are already staged or even committed can be "reverted" by replacing them with a version from the previous commit. Select a single resource in the Navigator or an analogous view and click **Replace With > Previous Revision**. The repository will determine the last commit that modified the selected resource and offer to replace the workspace resource with the contents of this commit.

This is mainly intended for "removing" single files from a commit (when committing the reverted workspace resources, they are effectively removed from the current commit). Even though this also works on folders and projects, the results of replacing a folder or project with a "previous revision" may be unexpected.

2.0.0 Git>Replace with Commit
Click **Replace With > Commit** to replace the selected files with their version corresponding to a selected commit.

2.0.0 Git>UI for patch creation
The UI for creating and applying patches has been added to Git functionality.

2.0.0 Git>git exclude
The git exclude function is now supported.

2.0.0 Git>Merge functionality
Support for Merge functionality has been added to Git functionality.

2.0.0 Git>Merge functionality improved
Merge functionality now includes the identification and resolution of Merge conflicts.

2.0.0 Git>Creating and applying patches
The UI and underlying functionality for applying creating and applying patches has been added to Git functionality.

2.0.0 RSEGit>Search for commit functionality
RSEGit Search for Commit functionality has been added to the Search menu in RSEGit.

2.0.0 RSEGit>Remote branch checkout
RSEGit now supports checkout of a remote branch.

2.0.0 RSEGit>Optimized to work with big projects
RSEGit was optimized to work with big projects

2.0.0 RSEGit>Drag-and-drop under RSEGit Staging view
Drag-and-drop of changes between Staged/Unstaged areas and “Open Working Tree Version” action under RSEGit Staging view is supported.

2.0.0 RSEGit>Synchronize workspace
The Synchronize Workspace function is now supported by RSEGit.

2.0.0 RSEGit>Synchronize View

The initial version of the Synchronize View in RSEGit has been implemented.

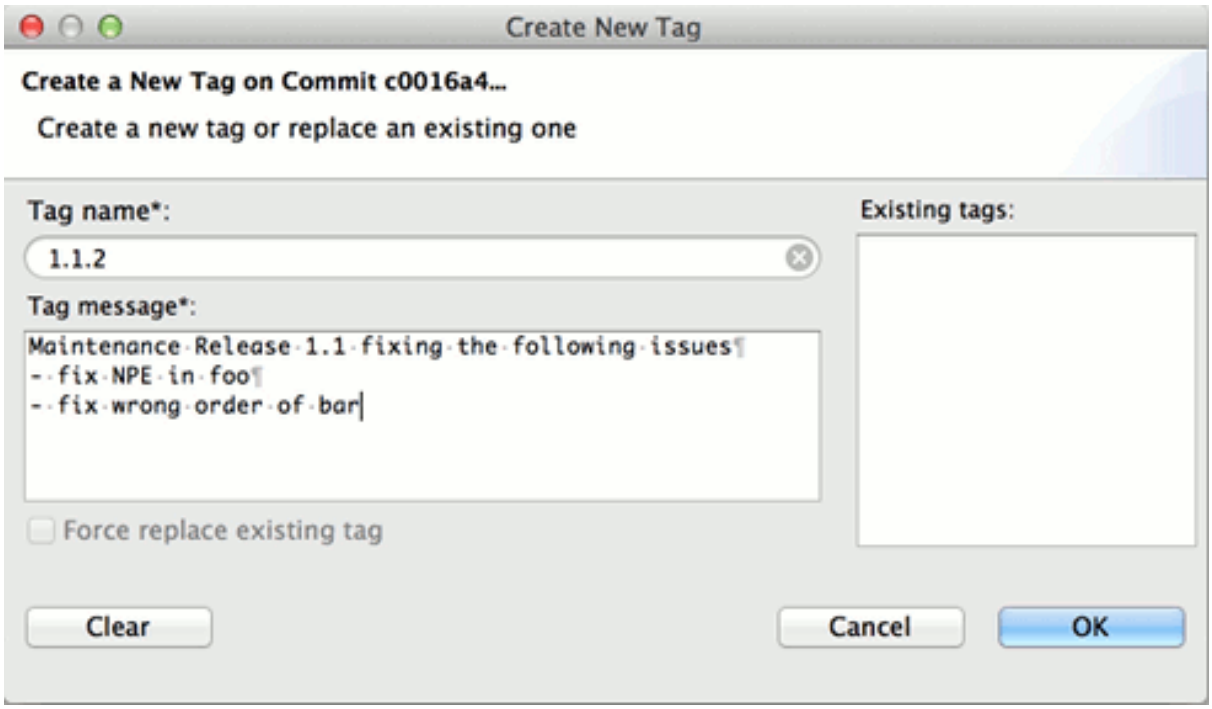
The menu command **Team > Synchronize Workspace** will launch the Synchronize View. This view allows you to inspect the differences between the resources in the local workspace and a local or remote tracking branch. Alternatively you may compare a local and a remote tracking branch. Comparison of two remote tracking branches as well as menu commands on the Synchronize View is not supported.

2.0.0 RSEGit>.gitignore

A `gitignore` file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected. For more details see Git documentation.

2.0.0 RSEGit>Add/delete tags for RSE Git

To Add a Tag :
Open the History View and click Create Tag... on the commit you want to tag



Enter the tag name
Enter the tag message
Click OK to create the annotated tag

Tags can also be created from the team menu, click Team > Advanced > Tag..., enter the tag name and message, select the commit you want to tag (default is HEAD) and click OK.

To delete a tag :
In order to delete a tag, select the tag to be deleted and click Delete Tag.

2.0.0 RSEGit>History View

The History View is now available in RSE Git.

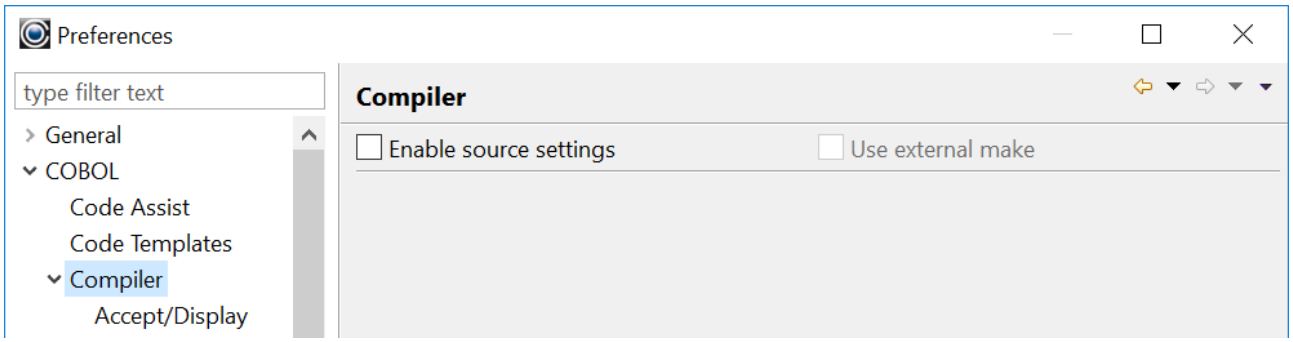
2.0.0 RSEGit>Compare View

The Compare View is now available in RSE Git.

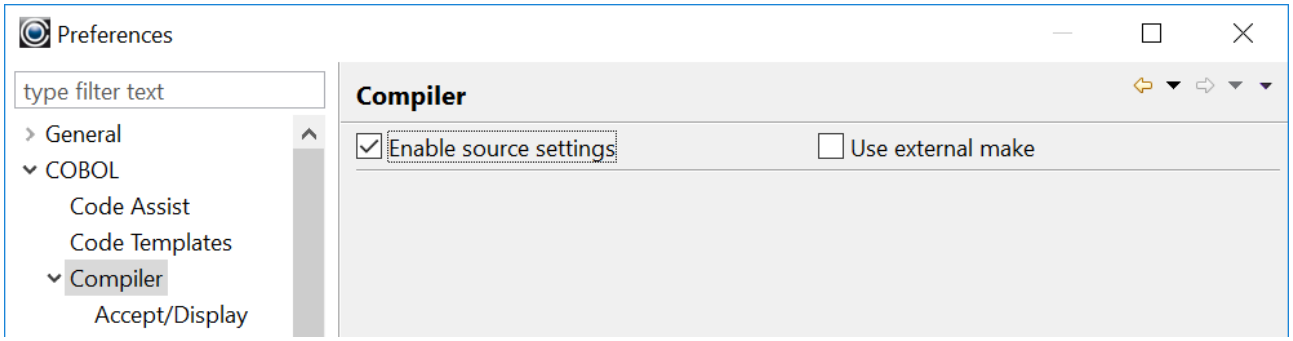
2.0.0 Compiler Options>Enable source settings/Use external make interface

“Enable source settings" and "Use external make" settings are displayed on each Compiler Options page. This enhancement allows the user to see controls disabled when Enable source settings is not set, and make the necessary modification on the same page.

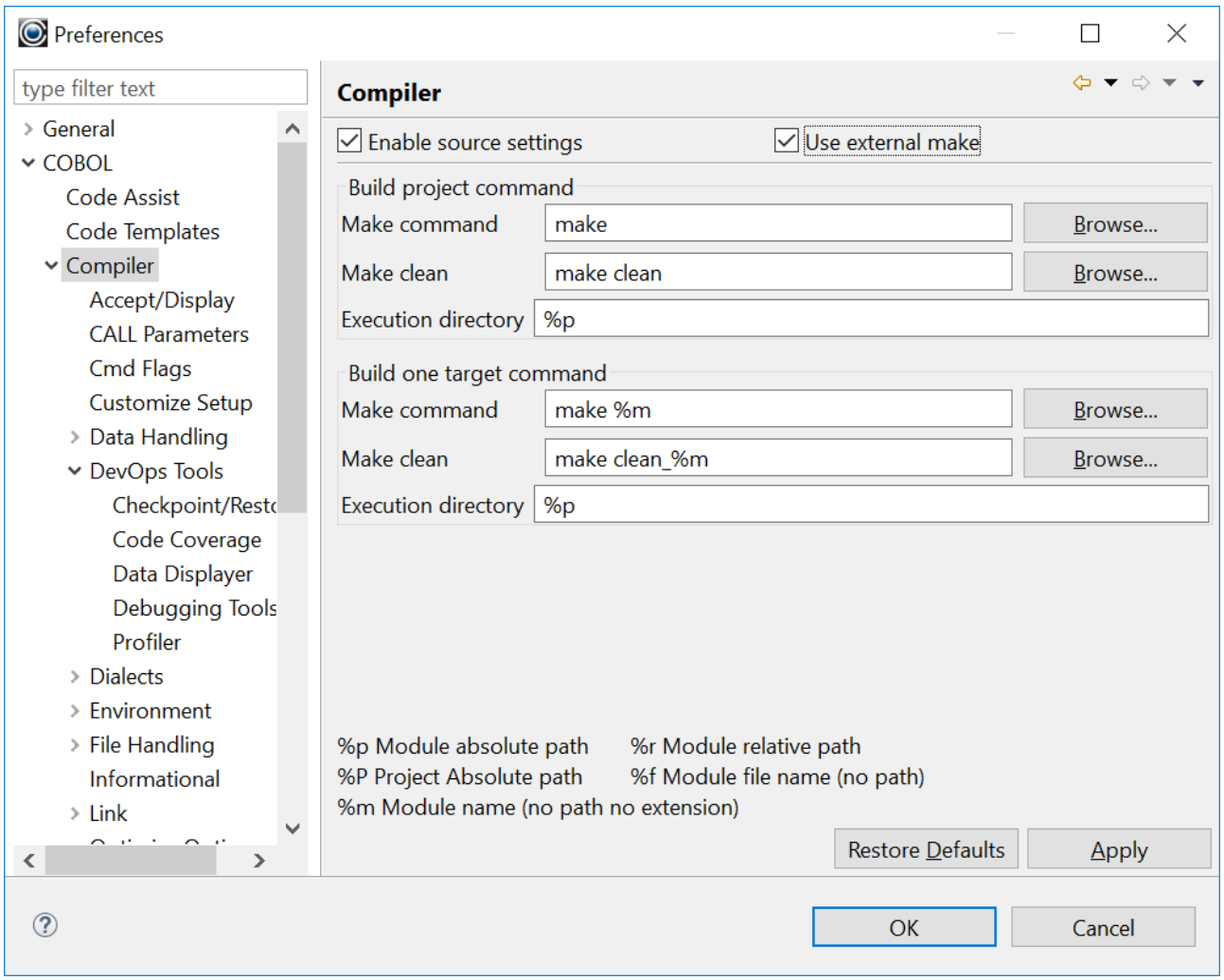
Note that when Enable source settings is not selected, the Use external make checkbox is disabled.



When Enable source setting is selected, the Use external make checkbox is enabled.

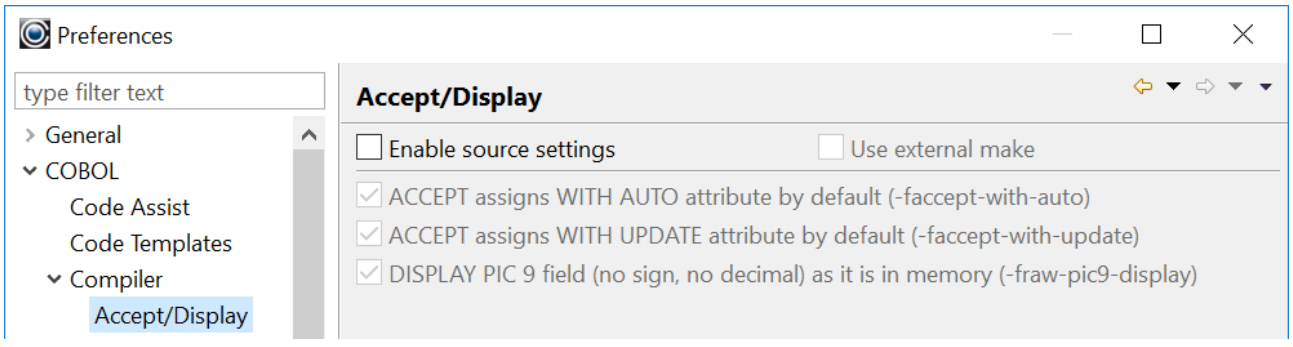


When the Use external make checkbox is selected, the External Make dialog is displayed. As long as the Use external make checkbox is enabled and selected, these options will override any settings that have been made on the Compiler Option Windows.

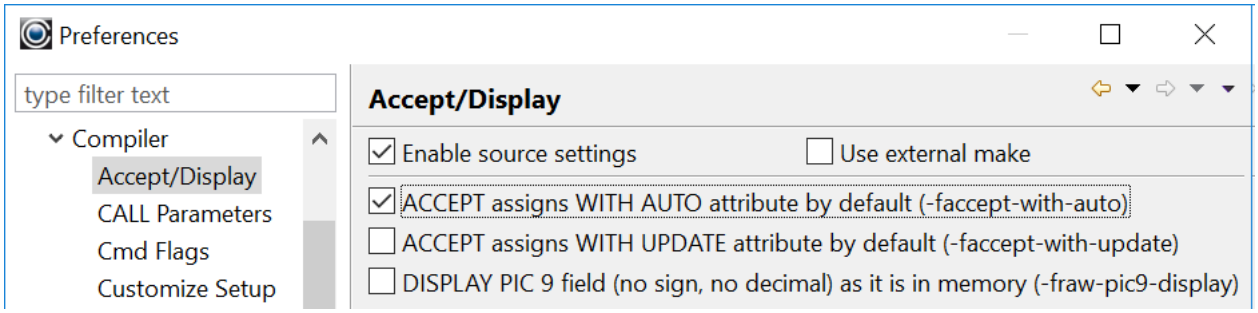


When navigating through the Compiler Option screens, you will see the state of the Enable source settings and Use external make checkboxes at the top of the screen.

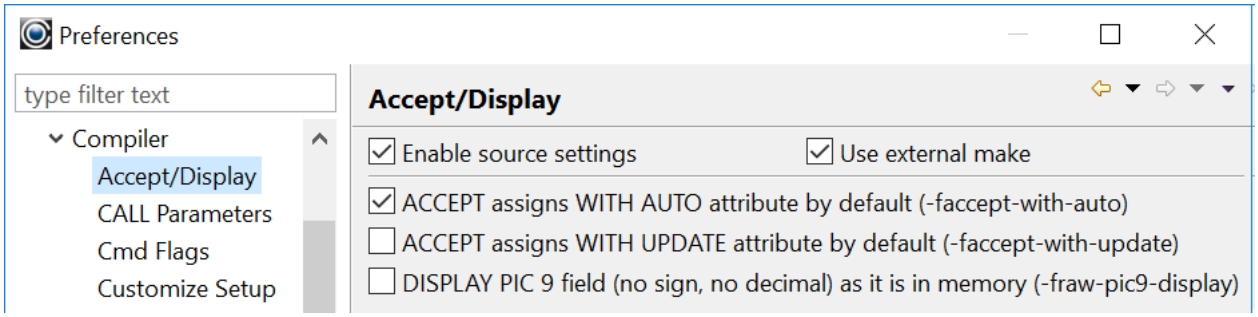
When Enable source settings is not selected, the controls on the screen are disabled.



When Enable source settings is selected and Use external make is not selected, the controls on the screen are enabled, and the settings made are incorporated into the next Build command.



When Enable source settings is selected and Use external make is also selected, the controls on the screen are enabled, and settings will be stored but will not be used until the Use external make checkbox is de-selected.



2.0.0 Compiler Options>External Make options open on Compiler page

Selecting the Use external make checkbox on the Window>Preferences>Compiler window opens the External Make dialog window.

2.0.0 RSEGit>History of changes for selected resource only not supported

Added ability to view history of changes for selected resource only

Fixes

2.0.0 RSEGit>Last commit not displayed.

In RSEGit, it was possible that the last commit would not be displayed in the History view.

This has been corrected.

2.0.0 Run/Debug anomalies on manually created run configuration

In some cases, manually created run configurations could fail to run correctly.

2.0.0 Tuned Developer Studio startup configuration

The Developer Studio startup configuration has been tuned to decrease loading time.

2.0.0 Compiler Options>Make Options renamed Project Options

Window>Preferences>COBOL>Compiler>Make Options have been renamed as Window>Preferences>COBOL>Compiler>Project Options.

2.0.0 RSEGit>Performance Improvements

RSEGit performance has been improved by caching Git objects.

2.0.0 Compiler Options>Added "Enable source settings" on the global level

The Enable source settings checkbox was not displayed on the Window>Preferences>COBOL>Compiler screen.

This has been corrected.

2.0.0 Profiler>Cleaned up charts when profiler session removed

When a profiler session was removed, the displays in the runtime tab were not removed.

This has been corrected.

2.0.0 Data Displayer>Fails to display a file

The Data Displayer could fail to display a file with File Error -1.

This has been corrected.

2.0.0 Opening project created with 1.8.x

Some workspaces created with version 1.8.x of the Developer Studio would not display correctly in the Navigator Window when opened with version 1.9.x.

This has been corrected.

2.0.0 Hovering Mouse over Variables in copy folder

ID# 1148768690

When variables were contained in a copy folder, it was not possible to hover the mouse over the variable display the variable value.

This has been corrected.

2.0.0 Code coverage>Does not decorate source code

Code Coverage would store information in the debugdb.dbd file, but when the runtime session was complete, the source code would not have decorated lines executed/lines not executed.

This has been corrected.

2.0.0 Platform-dependent setup scripts

(different on Linux/Windows, 32/64 bits)

It was possible for the setup script used in the Eclipse exec script that is automatically run before the makefile to incorrectly reference the location of the setup script file.

Example: `CALL %COBOLITDIR%\setenv_cobolit.bat`

This has been corrected.

2.0.0 Data Displayer>Cannot open ISAM file with Data Displayer

ID: 1148605328

When using Version 3.10.14 of the Compiler, the Data Displayer would display column headers, then fail to display the data, returning the error message: “An internal error occurred during “Get Data”.

This has been corrected.

1.8

1.8.11

New

1.8.10 Add ability to configure label breakpoint

ID#1147962810
You can now set a breakpoint on a label, such as a paragraph name, or section name.
From the command line debugger, if you want to set a breakpoint at a label called para-1:
>br para-1
>continue … The debugger will break at the para-1 label.

For example :

.0000010> main.
br para-1
Breakpoint 1 in para-1 at C:\COBOL\COBOLIT\Samples\hello.cbl
(cobcdb)
continue
-event-continue
(cobcdb)
.0000022> para-1.
-event-breakpoint-hit #0 hello () at C:/COBOL/COBOLIT/Samples/hello.cbl!22

From the Developer Studio, Debugger Perspective-

In the Debugger, select a label, right-click, and select the "Add Label Breakpoint" function from the dropdown menu. This creates a breakpoint, and selects (enables) it in the Breakpoint view.

Click on the Continue (F8) button.
The Debugger will break on the para-1 label.

1.8.10 Add ability to configure value breakpoint.

A value breakpoint is a breakpoint that cause the program to break after the value of a variable has been changed.

From the Developer Studio, Debugger Perspective-

In the Debugger, select a variable in the code editor, right-click, and select the "Add Value Breakpoint" function from the dropdown menu. This creates a breakpoint, and selects (enables) it in the Breakpoint view.

In the Variable View, select the variable, right click, and select "Watch" to add the variable to the Expressions view.

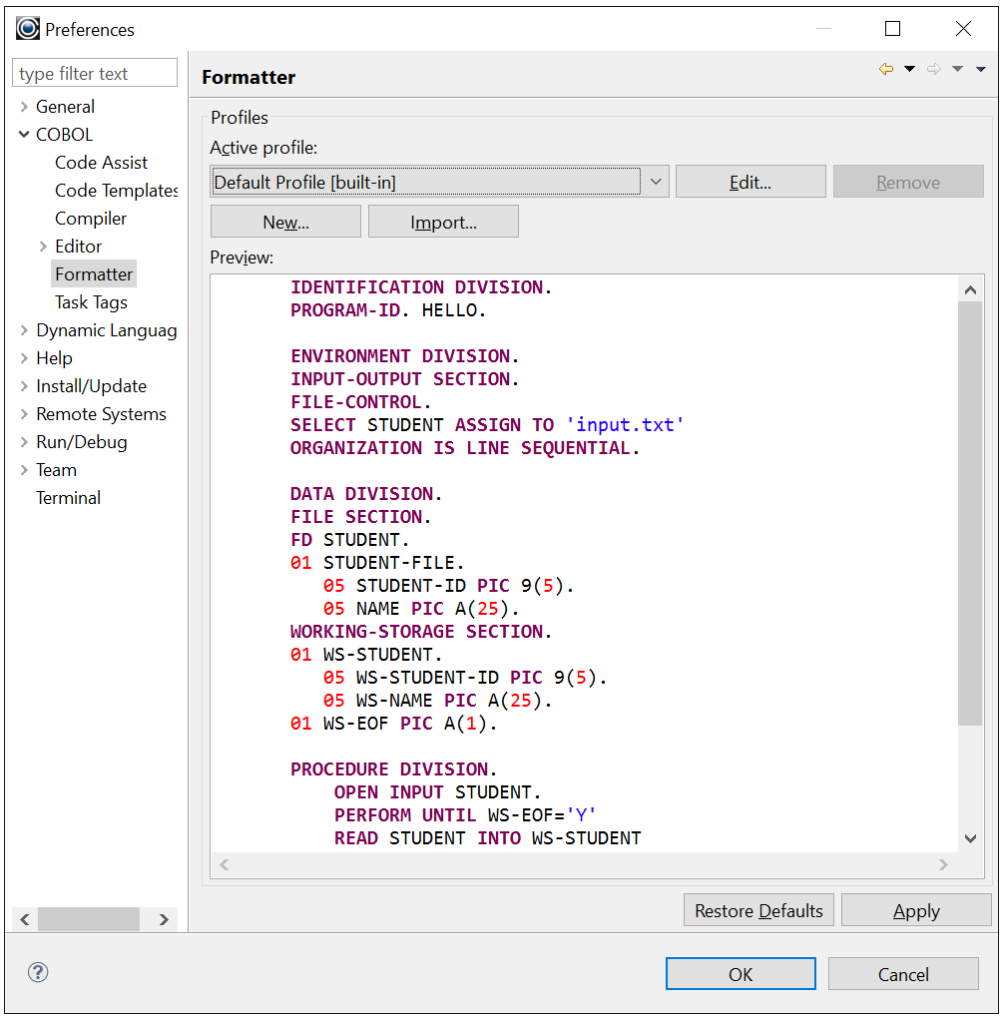
Click on the Continue (F8) button.
The Debugger will break after the variable changes value.

1.8.10 Update Developer Studio splash screen

The Developer Studio splash screen has been updated.

1.8.7 Window>Preferences>Formatter

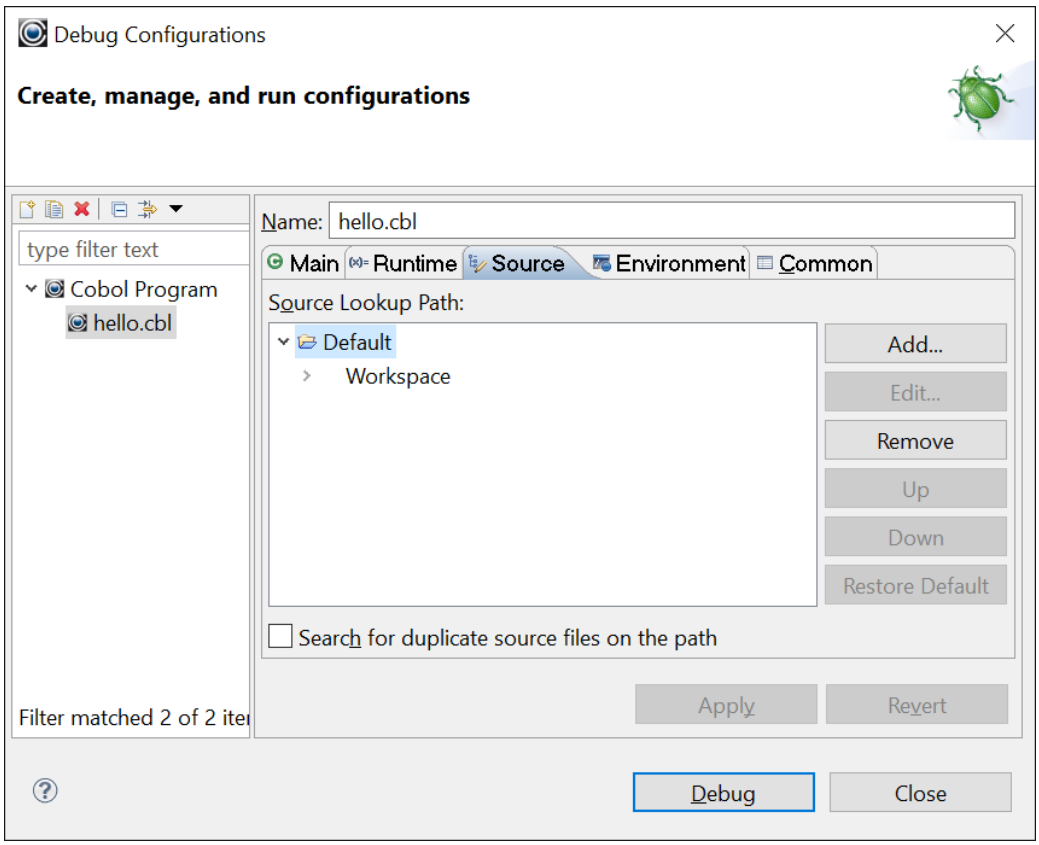
Window>Preferences>Formatter provides the ability to set formatting standards for your program.



1.8.7 Add Source location settings for debug configuration,

#1147276348

A Source tab has been added to the use current project as a first place to look for source files.
The Default setting is the current project folder. However, you can add and remove other locations.



1.8.7 Improve performance of Mark Occurrences and Rename Variable features.

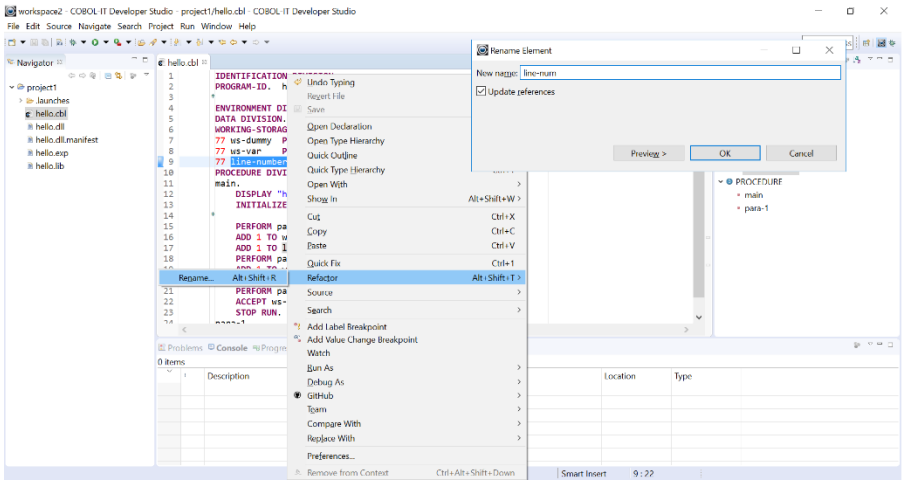
#1147428082

Performance has been improved on Mark Occurrences and Rename Variable (refactoring) features.

1.8.7 Refactoring

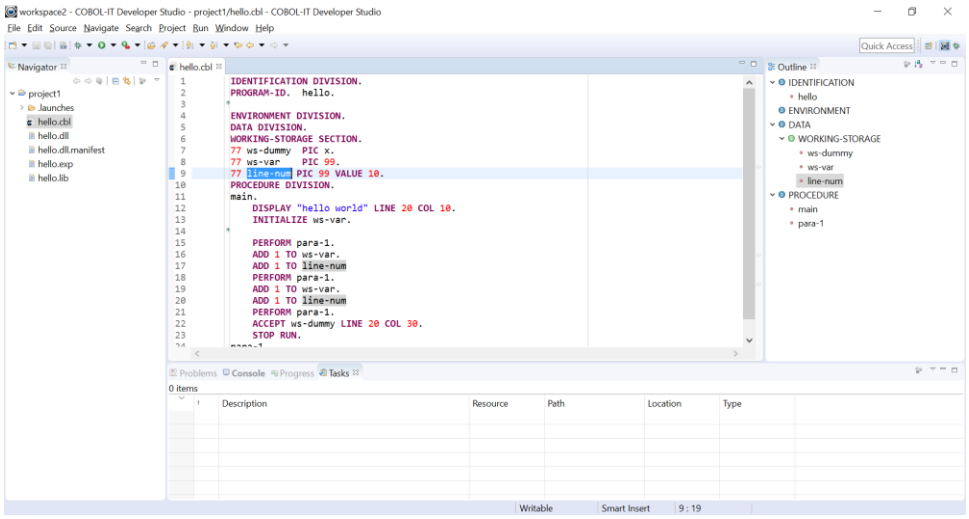
Refactoring provides the developer with the ability to change the name of a variable, and have all usages of that variable in the program changed.

In the example below, we select the line-number variable, right-click and select « Refactor ». The we select «Rename » from the subsequent dropw-down menu.



This opens the Rename Element window. Here, we change the name of the variable to line-num and click OK.

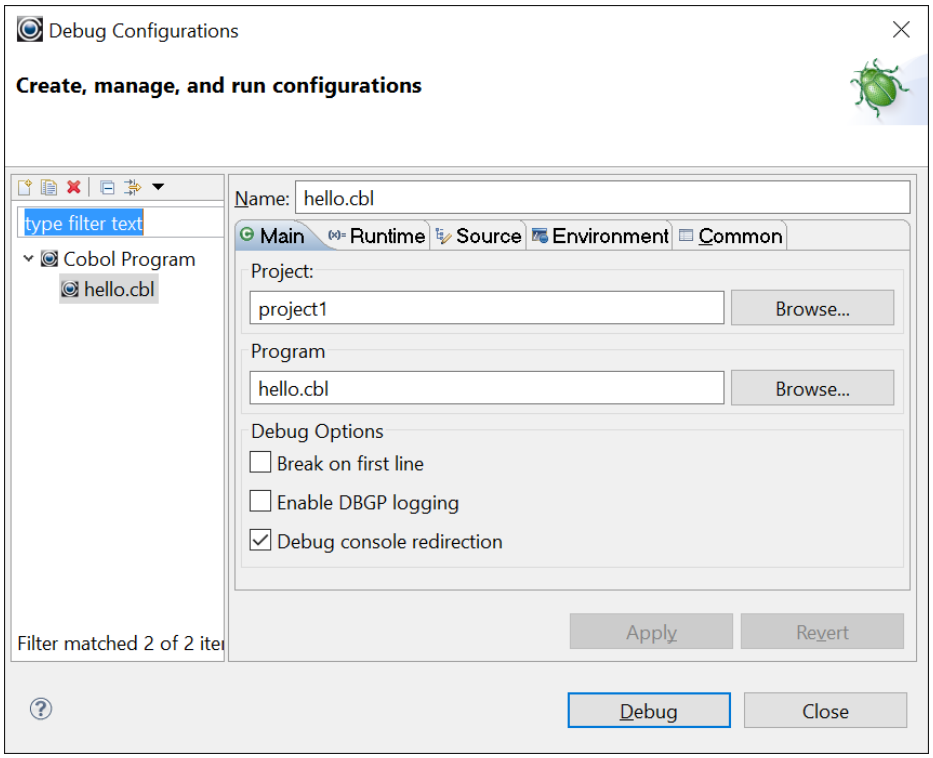
The Refactoring function causes the variable name to be changed, and all usages of the variable name in the program to be changed as well.



1.8.7 Debug console redirection

#1147323508

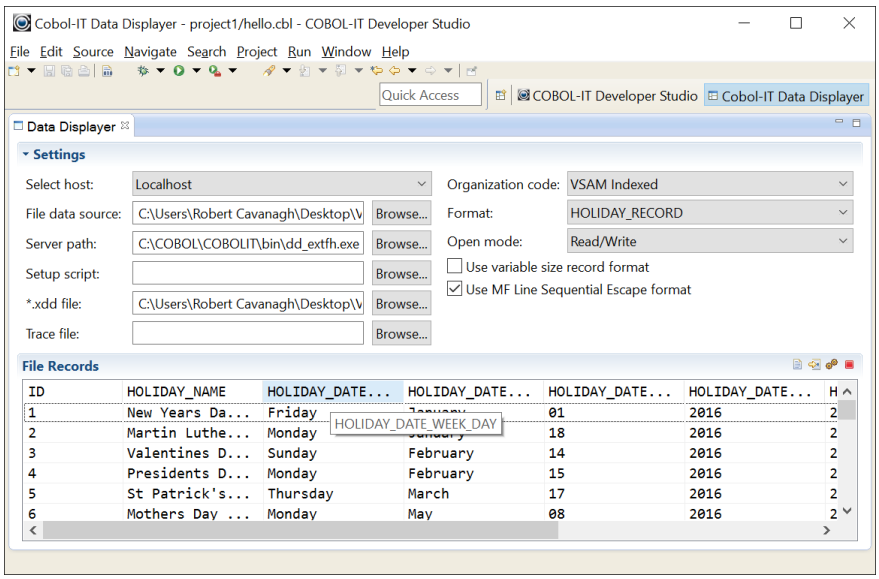
Debug console redirection is set by default on the Main tab of the Debug Configuration Screen. When debug console redirection is enabled, you can capture user input from the Console View while debugging. If the option is disabled, then user input can be ACCEPTed from the IDE's console in the Linux environment.



1.8.5 Add tooltips to column headers in Data Displayer.

#1147182780

For cases where a field name is longer than can be displayed in a column, hover over the column header to see the full field name.



1.8.5 Improve support for code templates in COBOL editor

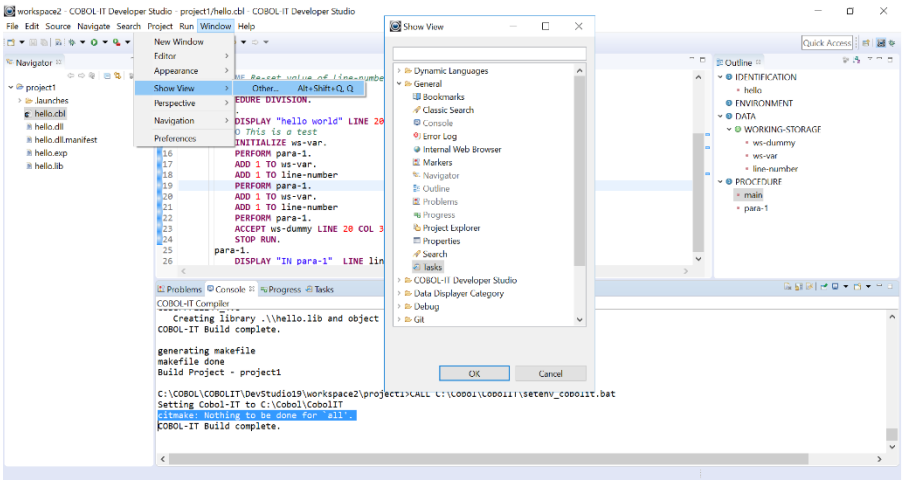
(add syntax coloring to template preview, use correct indent while inserting a template, second ctrl+space shows only templates).

1.8.5 Add Task Tags feature

The COBOL-IT Developer Studio contains a simple Task Manager, which consists of a Tasks View, and an interface for adding "Task Tags". After adding a Task Tag, the next time you build your program, the Task Tag will be added to the Tasks View. Click in the Tasks View to open the Task Interface.

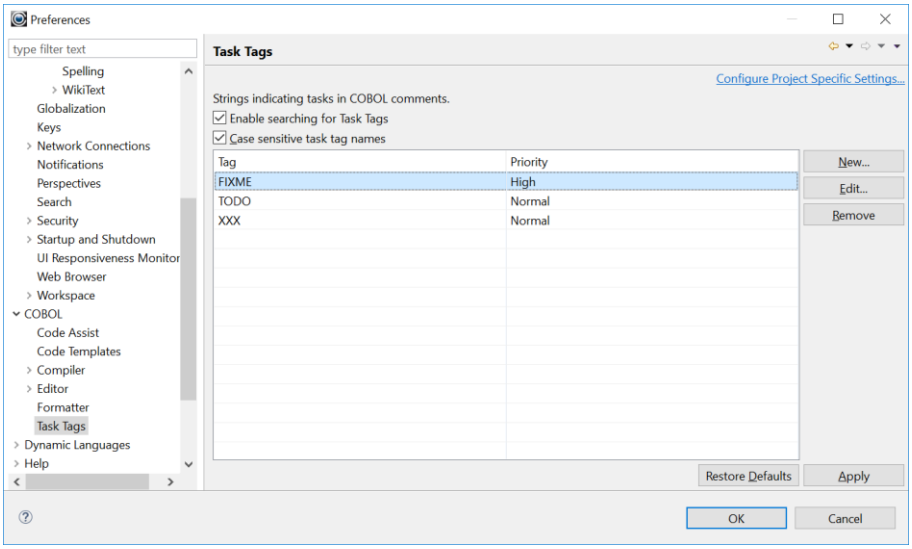
Showing the Tasks View

Use the Window>Show View>Other menu command to open the Show View Window.
From the Show View Windows, select General>Tasks. Click OK to open the Tasks View.



Window>Preferences>COBOL>Task Tags

Task Tags are strings indicating tasks in COBOL comments.
These include "FIXME" with a priority of High and TODO and XXX with priorities of Normal.



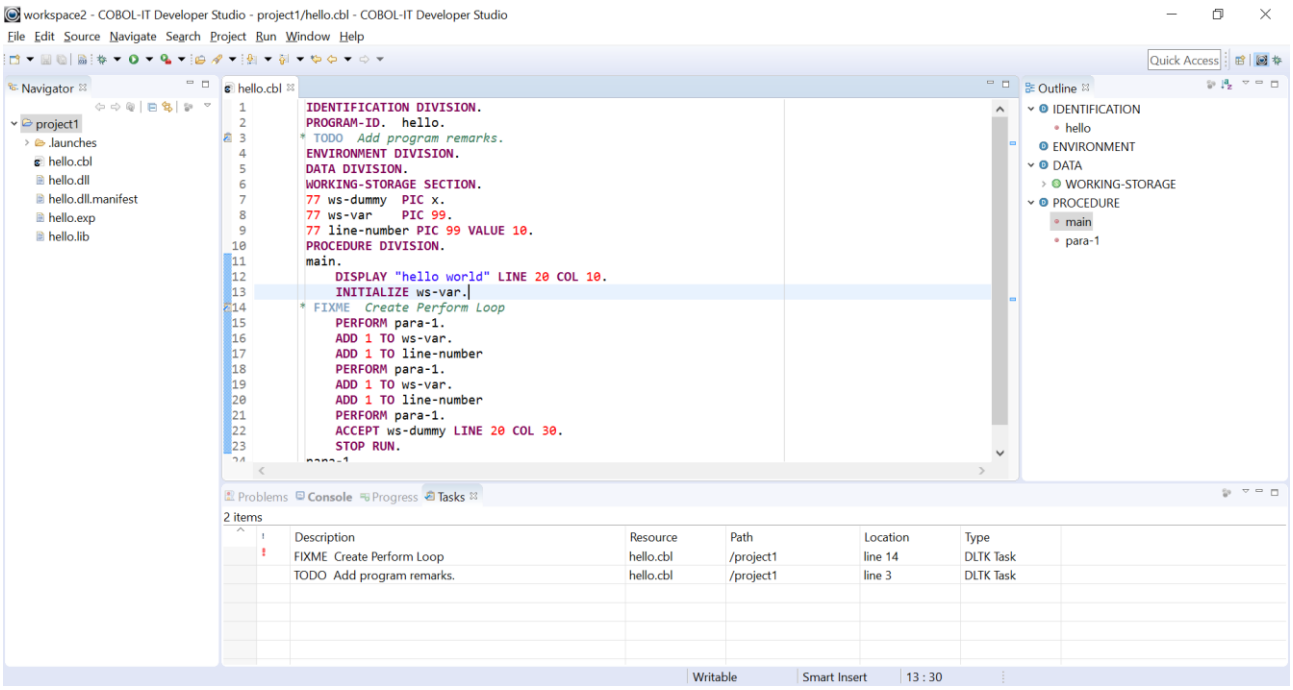
Adding a Task Tag to a source file

Task Tags are entered as Comments, followed by the Task Tag, and then a description.

In the example below, Task Tags have been added on line 3 and line 14. A decorator is added to the column to the left of the line number column when you add a task tag.

To add the task tags to your Tasks View, rebuild the project. The Task Tags in the Tasks View are sorted according to urgency. Items with a High Priority are sorted to the top of the list, and a ! decorator is added to the priority column.

Click on any column in a task line in the Tasks View to return to the location of that task in your program.



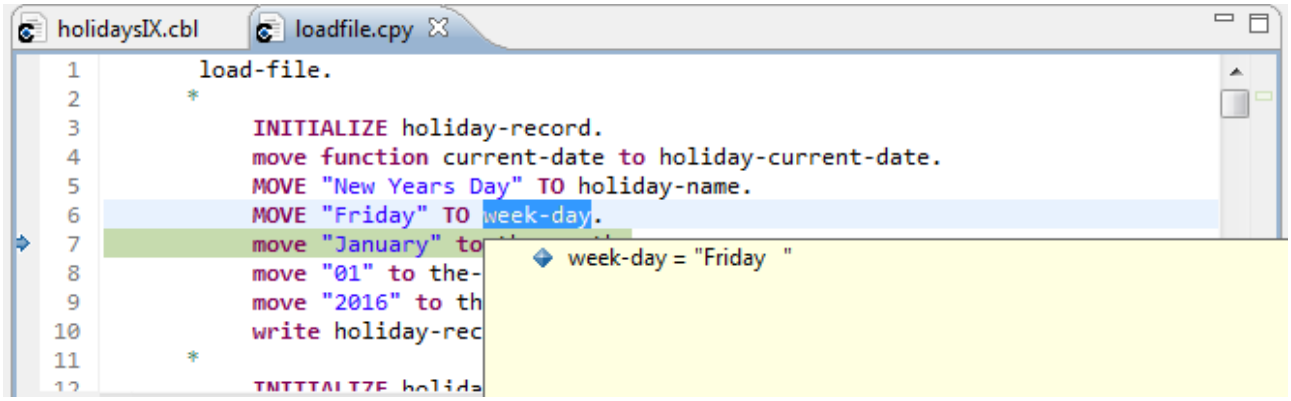
Removing a Task
To remove a task tag, simply delete the comment line containing the task, tag, save the source, and rebuild the project.

1.8.5 Add field picture to popup that appears on hover in debug mode.
#1147248370

1.8.4 Is bundled with Eclipse 4.3.2 and the JRE for both 32 or 64 bits
Beginning with version 1.8.4, the Developer Studio is bundled with Eclipse 4.3.2 (Kepler) and requires Eclipse 4.3.2 or a later version.

1.8.1 Evaluating expressions in Debugger supported in Windows.
The evaluation of expressions in the Developer Studio Debugger is now supported in all Windows environments. Requires Compiler Suite Version 3.9.13 or greater.

1.8.1 Show variable value hovering on variables in copybooks
ID #1147132570
When you hover over a variable in a PROCEDURE DIVISION copybook, it will now show the variable value.



1.8.1 Correct code style for "mfcomment" compiler option

ID #1147123260

The Developer Studio behavior now recognizes whether or not an asterisk in column 1 indicates a comment, based on the setting of the mfcomment compiler configuration file flag.

1.8.1 Fix hexadecimal value editing in the Expressions view

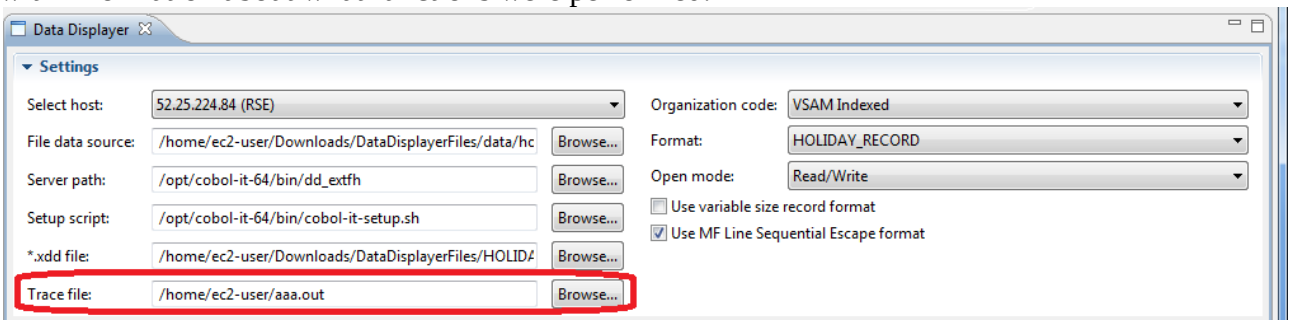
ID #1147059198

In the Expressions view, the "Edit hexadecimal value" function is now supported. Previously, the user would have to go to the Variables view to edit the value.

1.8.1 Add tracing for commands and data.

ID #1147054298

In the Data Displayer interface, to use a Trace file, browse to the directory in which you want the file created. You may either select an existing file or name your own. The file you select will be recreated each time you open a file in the Data Displayer. The trace file is a line sequential file, with information about what functions were performed.



The trace file should be a line sequential file. If it does not exist, it will be created.
Trace file information is renewed every time an Open function is performed.

Fixes

1.8.11 Ctrl+B reports « Nothing to be done » incorrectly

ID# 1147427810

After making a change to a copy file, and then closing and re-starting Developer Studio, the « Build » command (Ctrl+B) would report « Nothing to be done » incorrectly.
The information about the change to the copy file, and dependency that would require a build was stored in a temporary file which was removed when the Developer Studio was shut down. As a result, the information was lost.

This has been corrected.

1.8.11 cobcdb process not terminated when program has no 'STOP RUN'.

ID# 1148259316
When debugging a program in the Developer Studio, where the program has no STOP RUN statement at the end, the act of stepping through the last statement did not cause the cobcdb process to be removed from memory, and the process could claim high levels of CPU.

This has been corrected.

1.8.10 Adding table element to Expressions View

When adding a table element to the Expressions View, the index was not correctly transferred.

This has been corrected.

1.8.10 Debugging session not terminated correctly

In some situations, a debugging session was not correctly terminated.

This has been corrected.

1.8.10 Program Structure incorrect with EXEC statement

In some situations, in programs containing EXEC/END EXEC statements, the program structure could be mis-constructed.

This has been corrected. For purposes of representing the program structure, everything inside and EXEC statement is ignored.

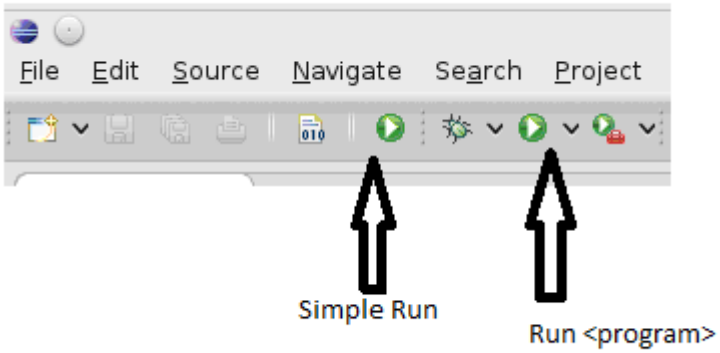
1.8.9 Improve debugger stability

Pressing the « Step Into » hotkey and holding it down could cause the debugger to abort.

This has been corrected.

1.8.7 Hide "Simple Run" toolbar item from linuxtools for dev studio perspective.

#1147323078
The « Simple Run » toolbar item in the Developer Studio Perspective displayed in the Linux/UNIX installation, but did not function.



This has been corrected. The « Simple Run » toolbar item has been removed from the toolbar.

1.8.7 Regenerate cobolit.makefile during build if the file doesn't exists.

#1147427868
If the cobolit.makefile did not exist, the build process could stall.

This has been corrected. The cobolit.makefile will be regenerated if it does not exist.

1.8.7 IndexOutOfBoundsException

Invoking code completion in an empty file could create an Index Out of Bounds Exception.

This has been corrected.

1.8.7 Correct Outline View source parser

#1147461076
When data structures continued past column 71 (to column 72), the outline view would not represent the data structure correctly.

This has been corrected.

1.8.7 Debugger anomaly after editing variables in Expressions View

#1147387842
If, while debugging, you selected « Edit hexadecimal value » in the Expressions View, and then switched to Text Mode, and entered a new value, the debugger could lose track of subsequent breakpoints.

This has been corrected.

1.8.5 Add missed JRE 6u45 binaries.

The JRE 6u45 binaries were not included in the 1.8.3 download of Developer Studio.

This has been corrected.

1.8.5 Data editing anomaly in Data Displayer after columns configuration

#1147308902
If you hid a column from view in the Data Displayer then attempted to "Edit hexadecimal value" on a field further along in the record, you would see the value from the wrong field.

This has been corrected.

1.8.5 Fix default code templates availability on linux and mac platforms

The complete set of COBOL code templates was not available in Window>Preferences>COBOL>Code Templates on the Linux and Mac installations of Developer Studio.

This has been corrected.

1.8.5 Hexadecimal edit in Expressions view.

#1147215380
The hexadecimal edit feature in the expressions view was not activated.

This has been corrected.

1.8.5 Active state of COBOL editor not preserved after re-starting Eclipse

#1147243756
It was possible that when Windows>Preferences>General>Editors>Restore Editor State on Startup was selected, in some cases, the Editor would retain the same focus that was on the editor at shut-down after the subsequent startup.

This has been corrected.

1.8.5 Debugger aborts when F5 is pressed too fast

#1147243684
It was possible to cause the Debugger to abort when the F5 (Step Into) key was pressed too fast.

This has been corrected.

1.8.1 Clean up old temporary files in new eclipse session

ID #1147123450
The eclipse workspace directory would not clean up all temporary folders created and used by the COBOL-IT debugger and runtime. In some situations, this could cause problems.

This has been corrected.
Old temporary folders and files are now cleaned up when beginning a new eclipse session.

1.8.1 Fix target position for GoTo action in DataDisplayer

ID #1147054522
The GoTo Line number function could cause the Data Displayer to move to the next line after the line indicated by the line number.

This has been corrected.

1.7

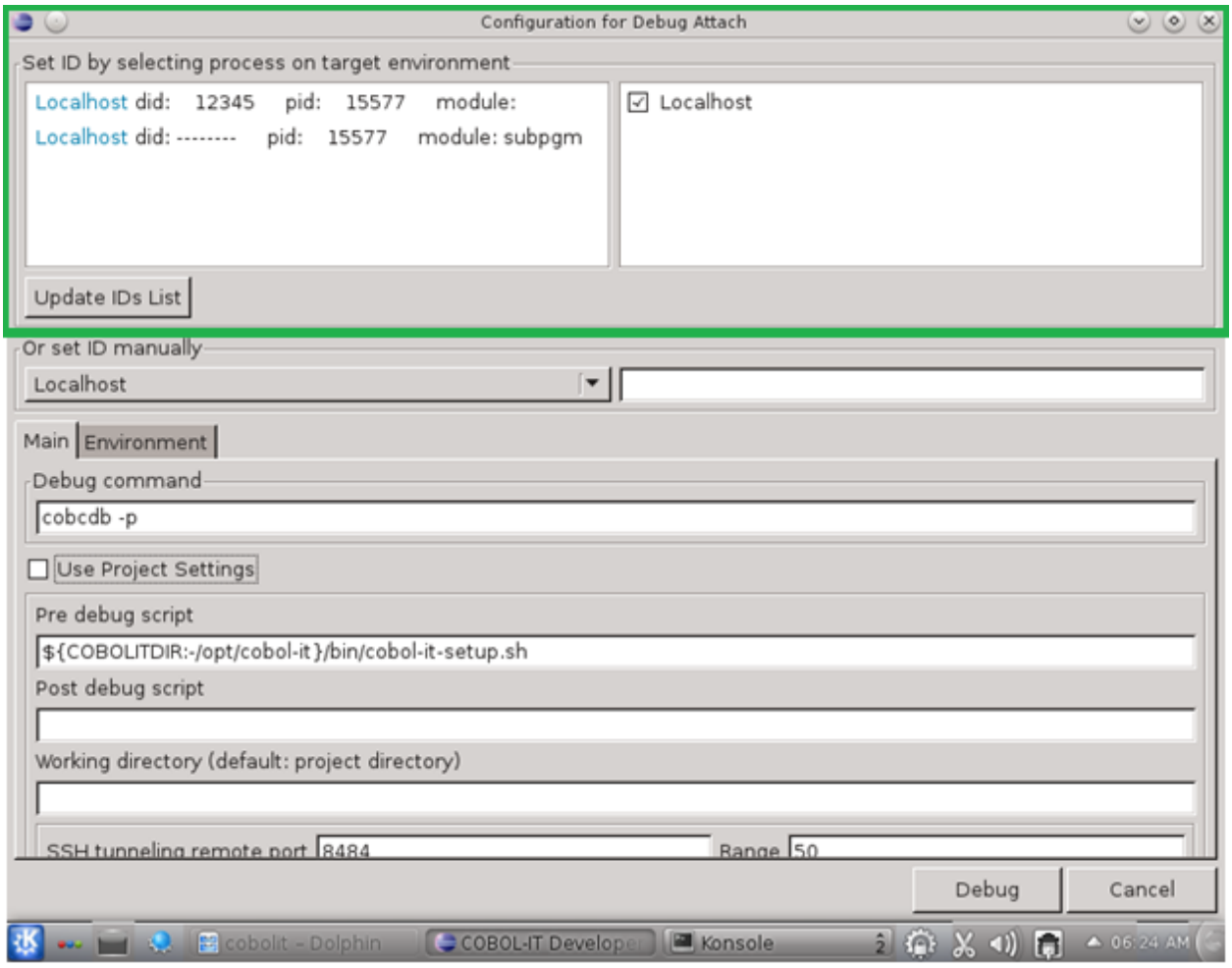
1.7.21

New

1.7.21 Debug Attach Interface

The “Debug Attach” interface has been updated.

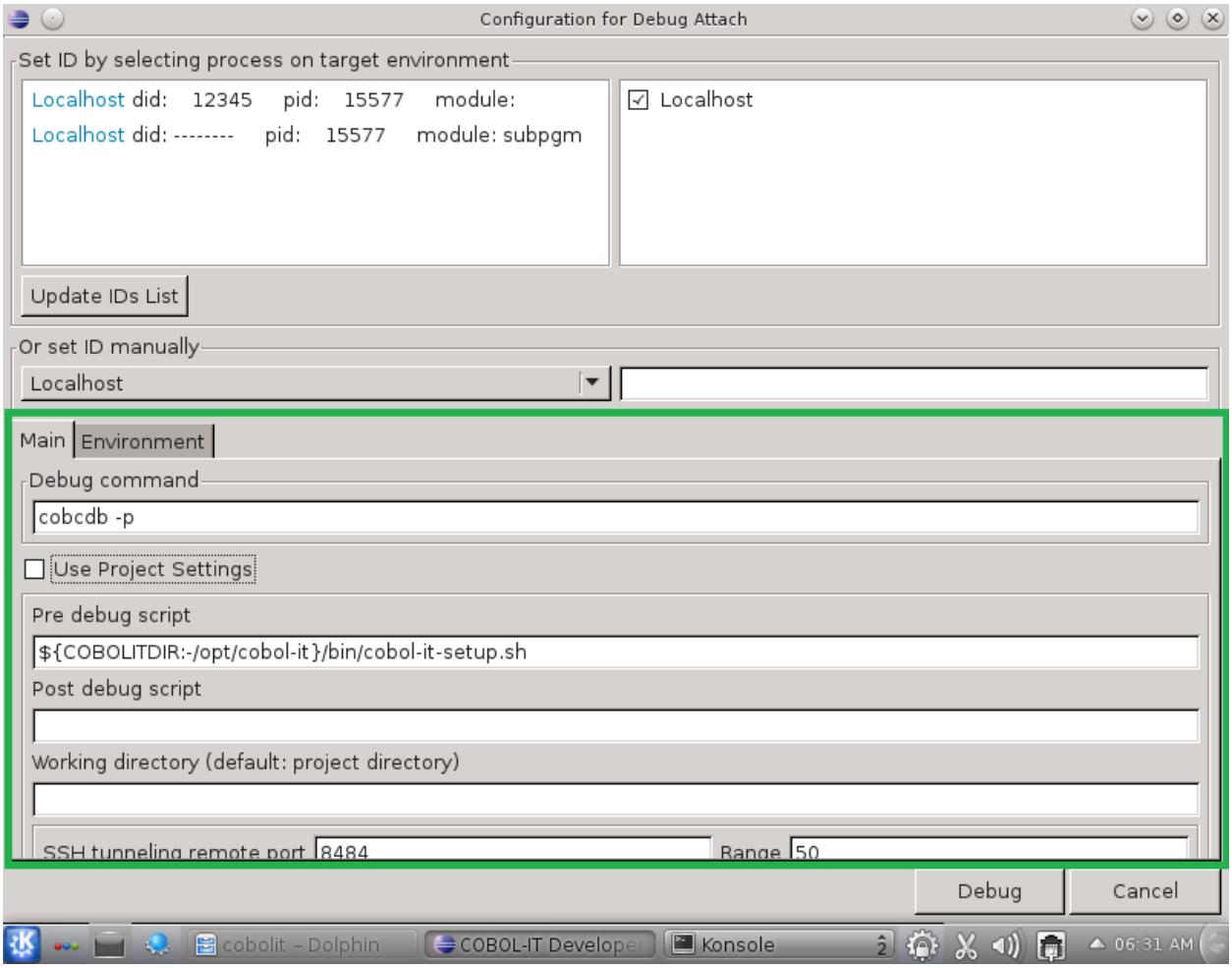
The « Debug Attach » interface is divided into two parts. The upper part of the screen provides functionality to set the debug-id. The « Set ID » interface allows you to select the process-id of a running COBOL program, or to set a debug-id manually.



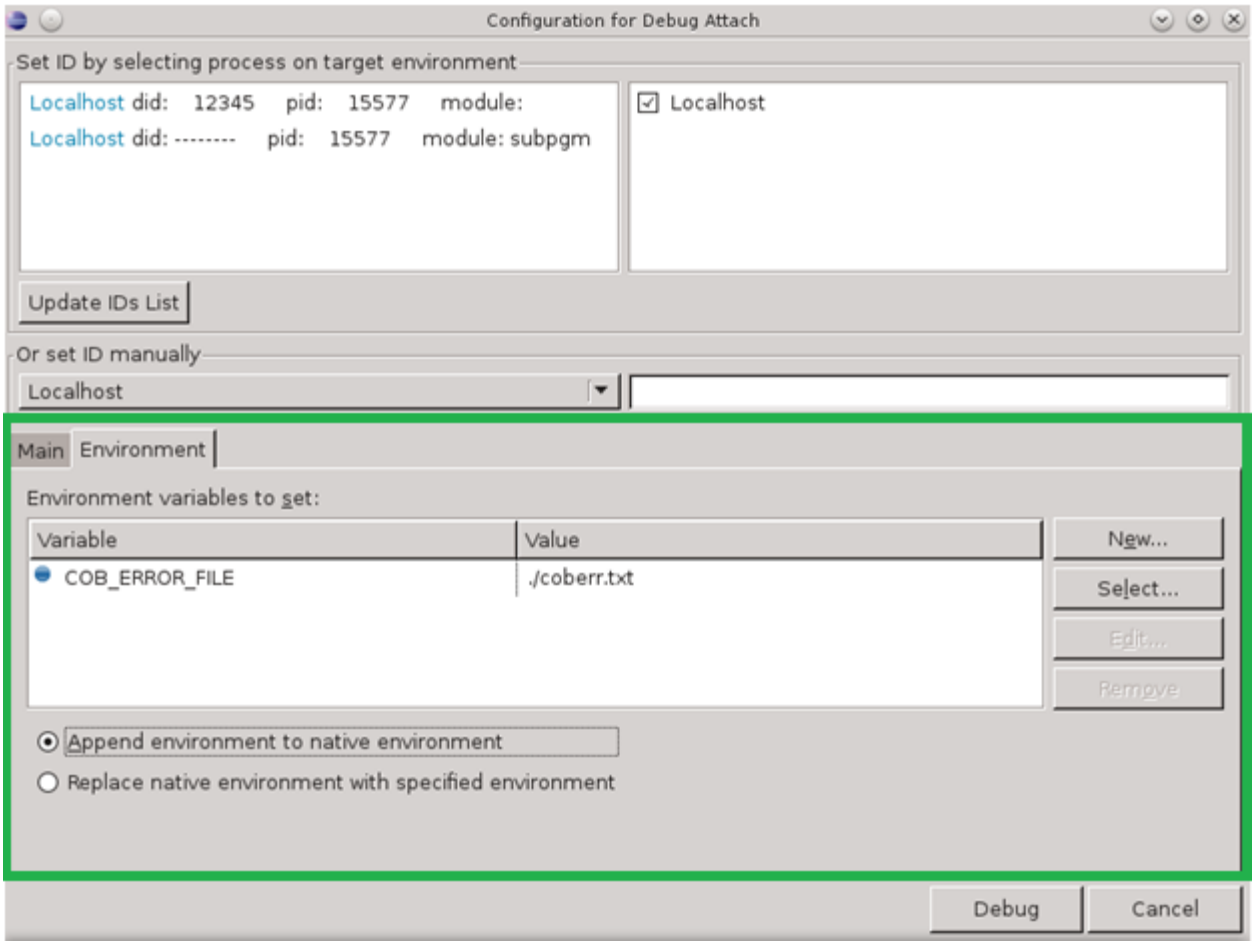
The “Set ID” interface all defined connections, including both the Localhost, and the Remote Systems to which the Developer Studio is attached.

Select one or more of these connections by clicking on the connection. For each connection that you select, the Developer Studio will scan that connection for running COBOL programs. You can then select the target that you wish to debug.

The lower part of the screen provides « Main » and « Environment » tabs which allow you to create a Debug Configuration. On the « Main » tab, you can name a « Pre-debug », and « Post-debug » setup script. You can name your Working Directory. By default, this will be your Project Directory. If you have a remote connection, you can configure specifics for the SSH tunneling-related variables.



On the « Environment » tab, you can set environment variables that you find useful for debugging purposes. In this example, we have set COB_ERROR_FILE to coberr.txt.

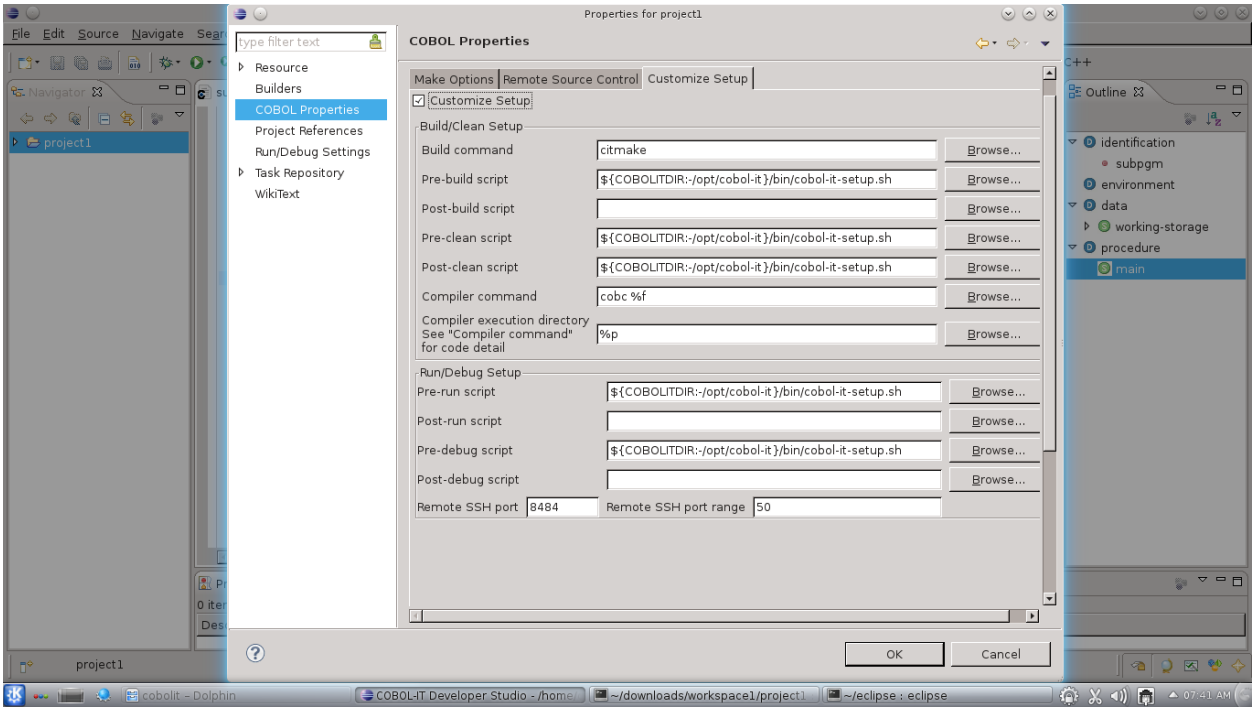


After you have entered the debugger, the “Debug Attach” interface requires input from the user to locate the source files that are associated with the running COBOL programs.

Note that when locating the source file that is associated with the running COBOL program, it is no longer a requirement that the source file be located on the same server as the running COBOL program. There is a limitation to this rule- when operating on multiple servers, the COBOL-IT setup script must have the same location on all servers.

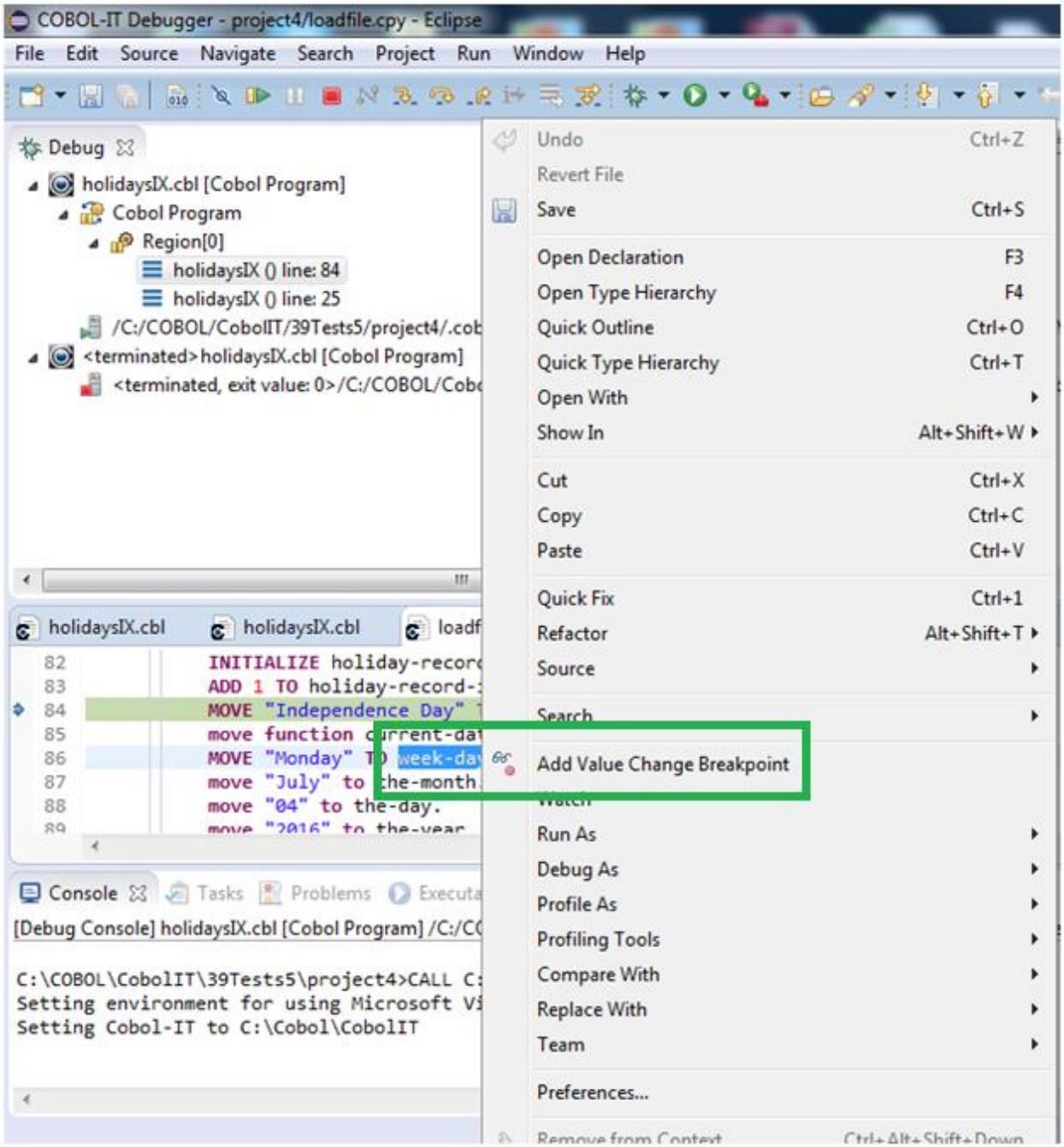
1.7.21 Custom Setup Interface

The “Custom setup “ Cobol properties to customize setup script includes the interface for setting SSH ports and is now presented on a separate tab titled “Customeize Setup” on the COBOL Properties window.

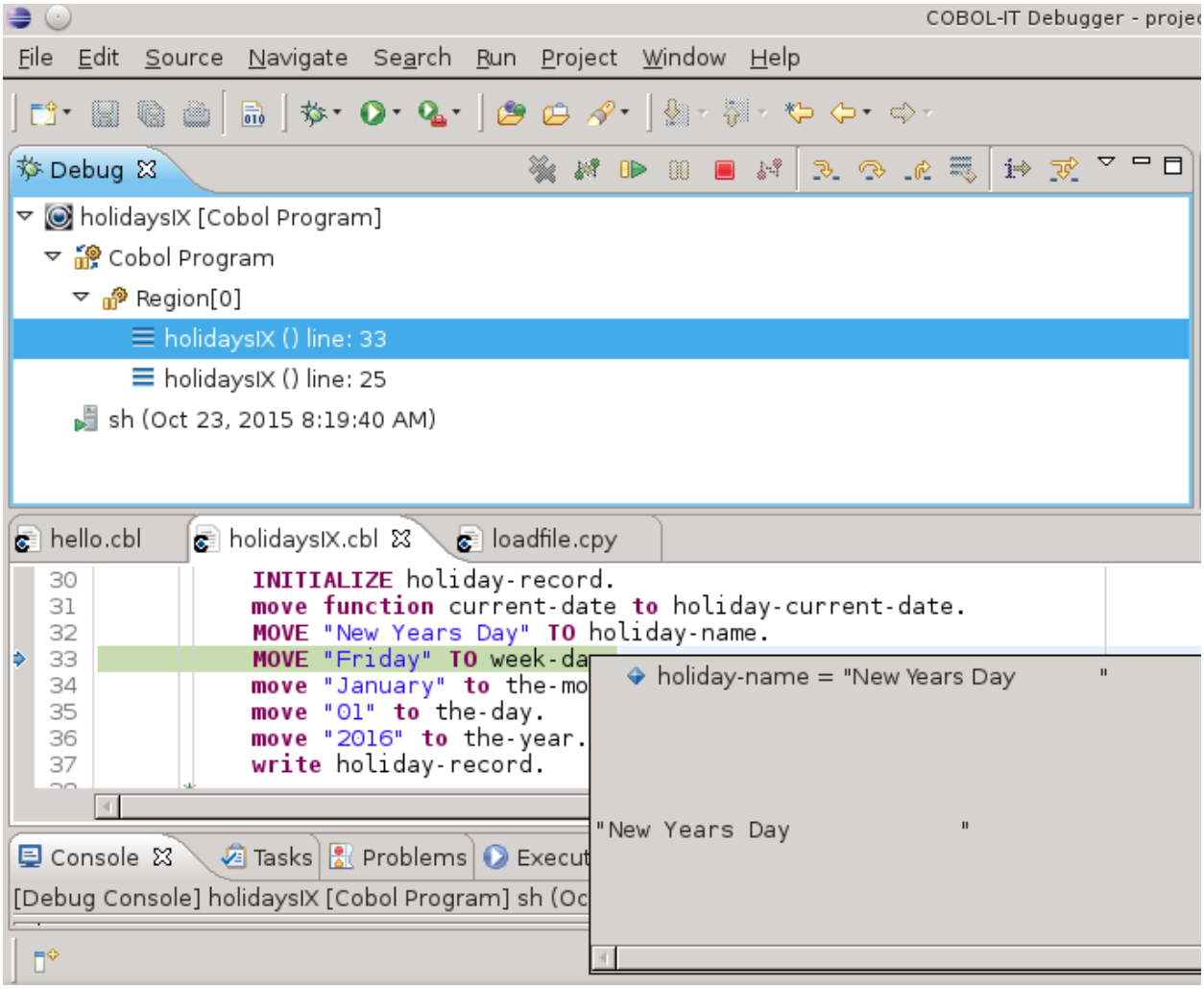


1.7.21 Add Value Change Breakpoint from right-click dropdown menu

Select a variable in the Debugger editor. Right click on it, and select "Add Value Change Breakpoint" from the dropdown menu. This will create a new "Value Change Breakpoint". Highlight the variable in the variable window, and the debugger will break whenever the variable changes value.

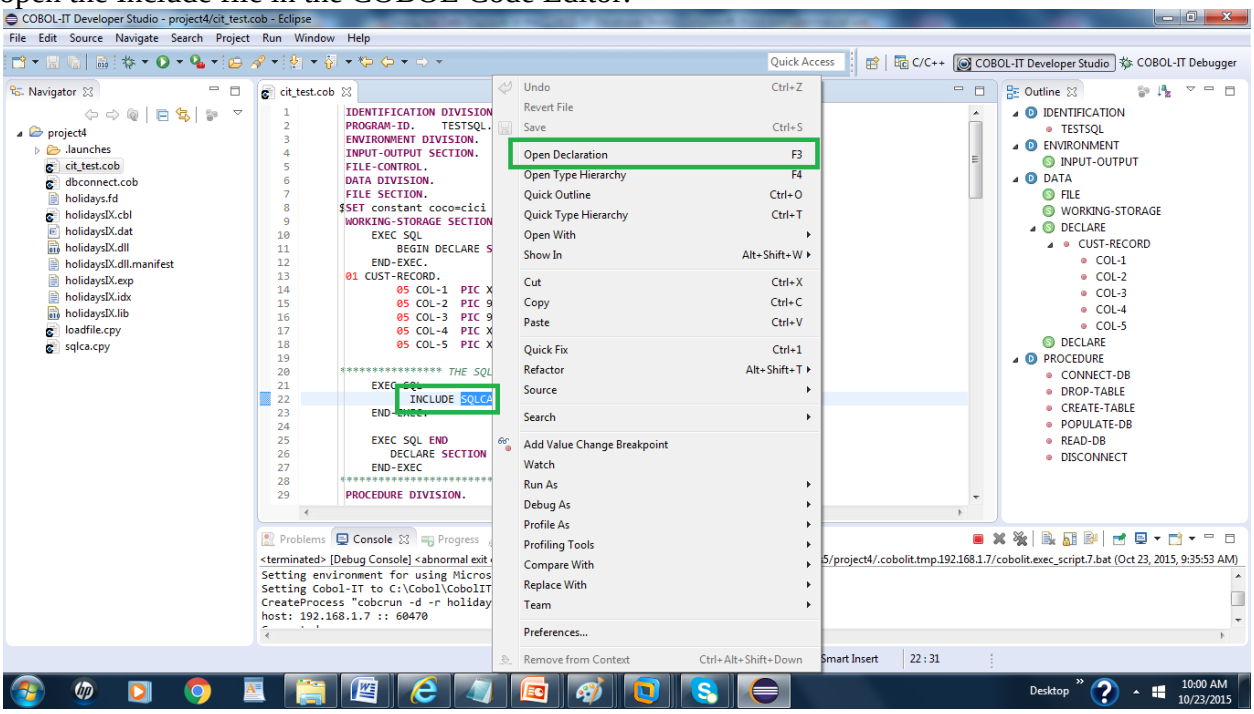


1.7.21 Hovering over variable returns value

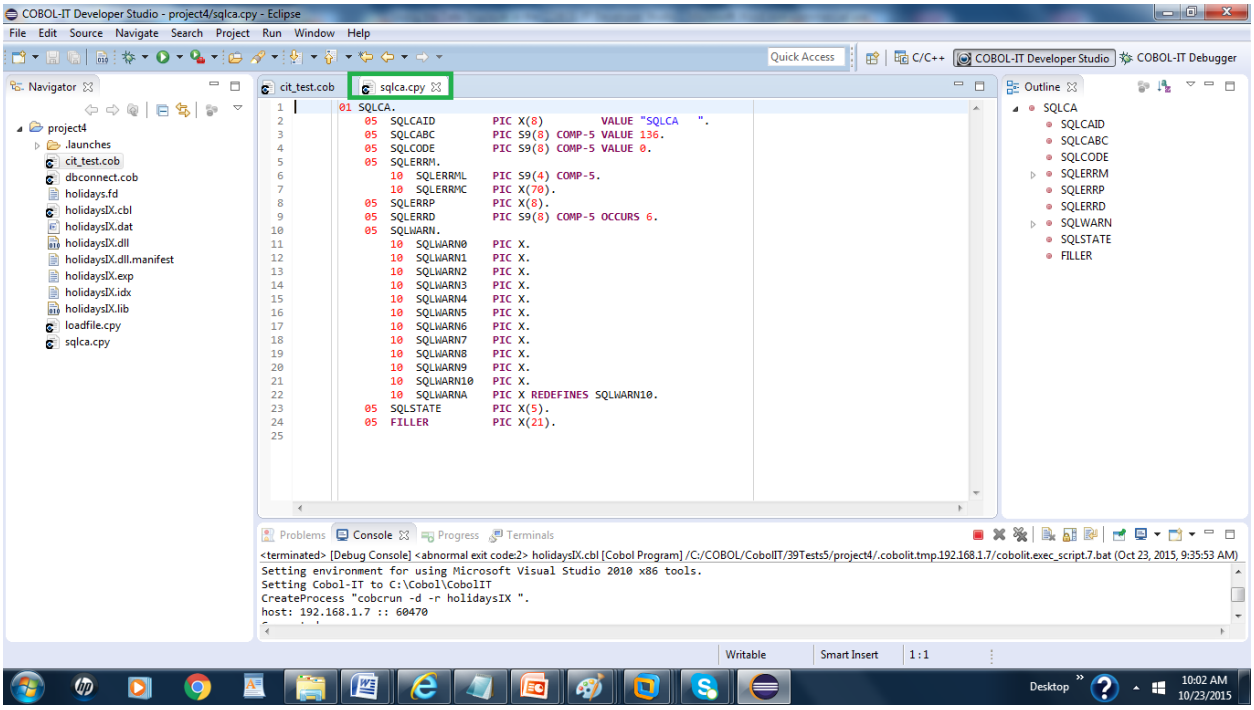


1.7.21 F3 on SQL Include open the include

Select an “include” file in your source file, and right-click on it. Select Open Declaration (F3) to open the Include file in the COBOL Code Editor.



SQLCA opens in the COBOL Code Editor.



1.7.21 Data Displayer Interface

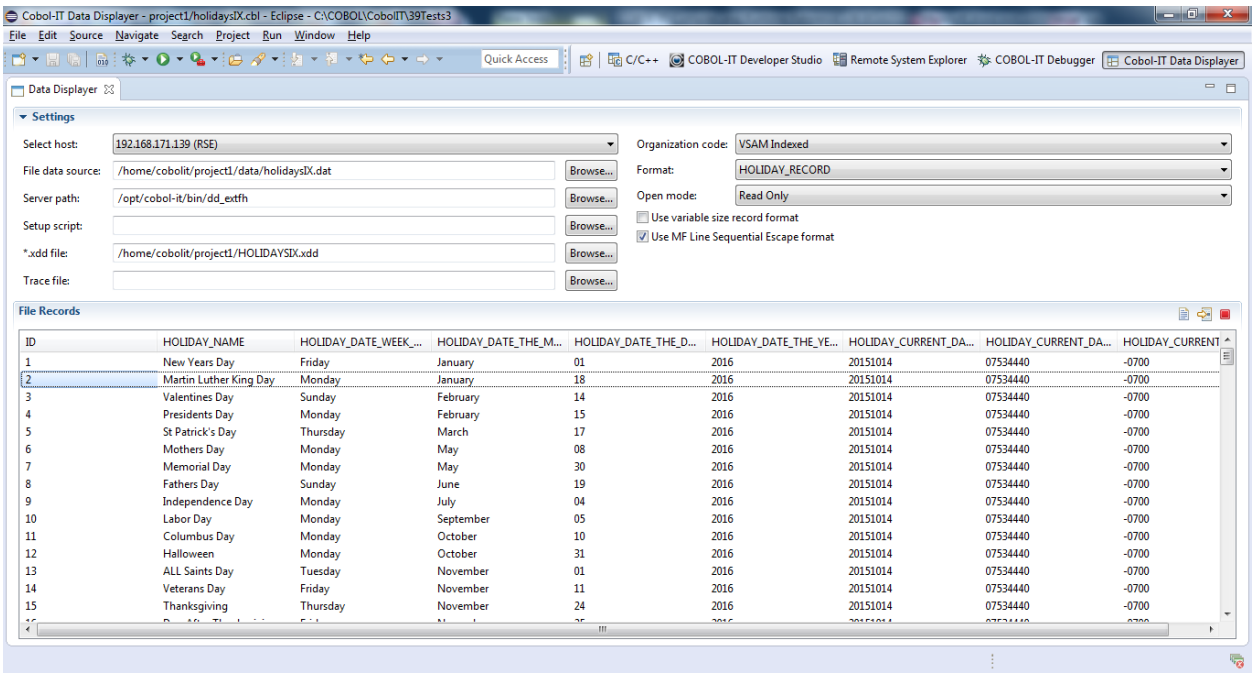
Using the Data Displayer in the COBOL-IT Developer Studio

The COBOL-IT Data Displayer uses an EXTFH server, and Data Dictionary files (**XDD**s) to interpret and display data files.

The EXTFH server, dd_extfh, is included in the bin directory of the COBOL-IT 3.9.x distribution. To generate an XDD, compile a COBOL program that includes the FD for your file, including the –fgen-xdd compiler flag.

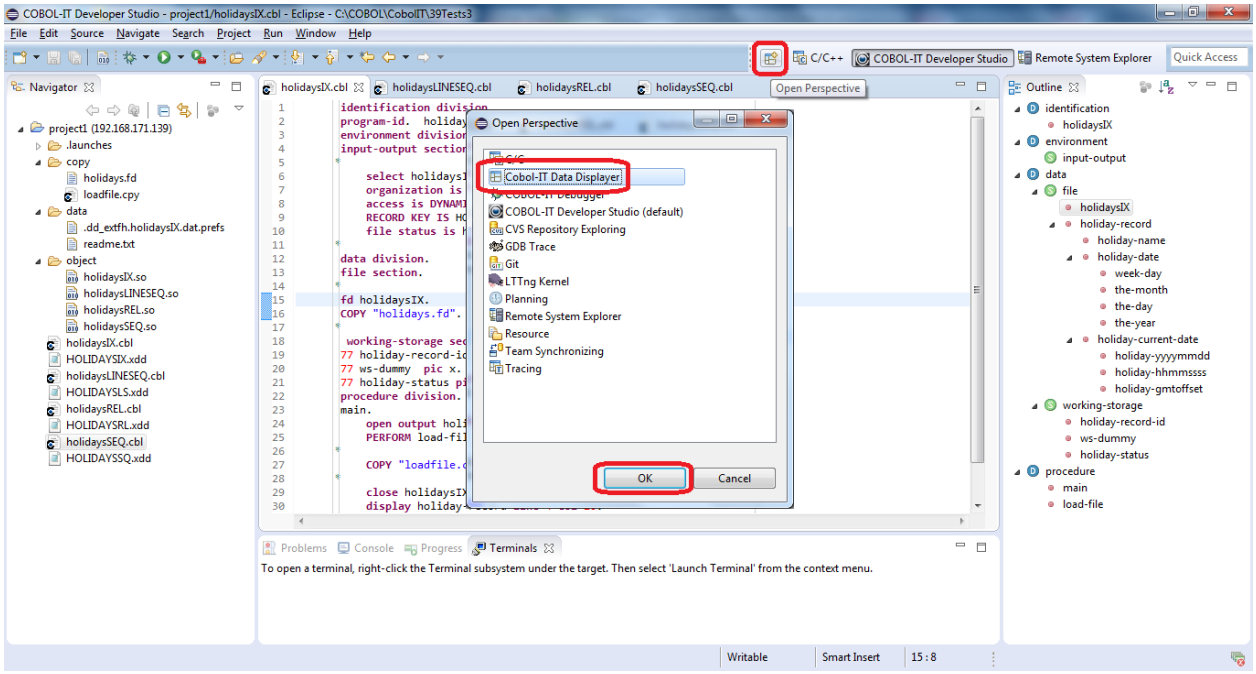
The File Records Toolbar of the COBOL-IT Data Displayer provides toolbar buttons that allow you to Open a file, Go to a specific record, and Close a file.

The Data Displayer provides an interface in which Data File can be OPEN'ed for Read-Only, or Read-Write. Data is READ a page at a time. When OPEN'ed for Read-Write, elements of a record can be edited in ASCII, or Hex.



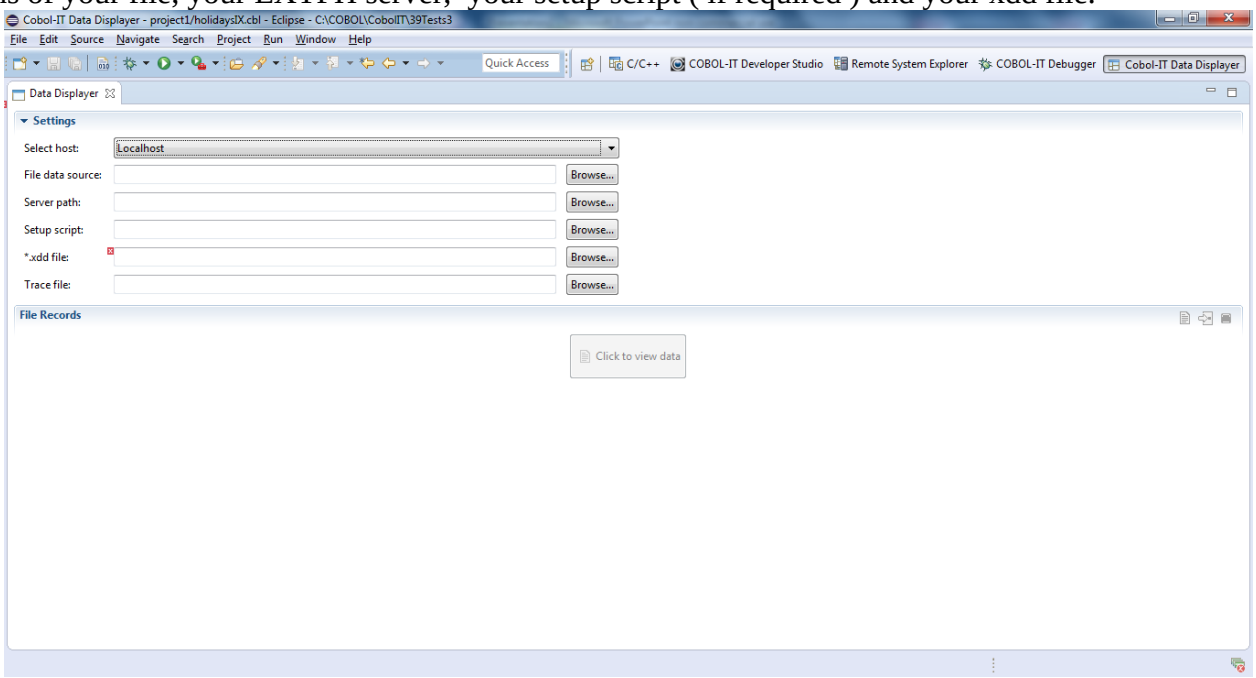
Open the Data Displayer Perspective

To open the Data Displayer, Click on the Open Perspective toolbar button. In the Open Perspective Window, select COBOL-IT Data Displayer. Click OK to open the Data Displayer Perspective.



The Data Displayer interface

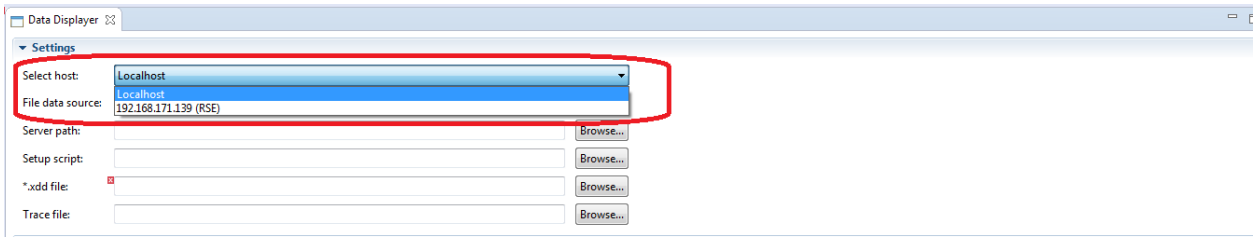
When you first open the Data Displayer, the interface allows you to Select the host, and identify the locations of your file, your EXTFH server, your setup script (if required) and your xdd file.



Select Host

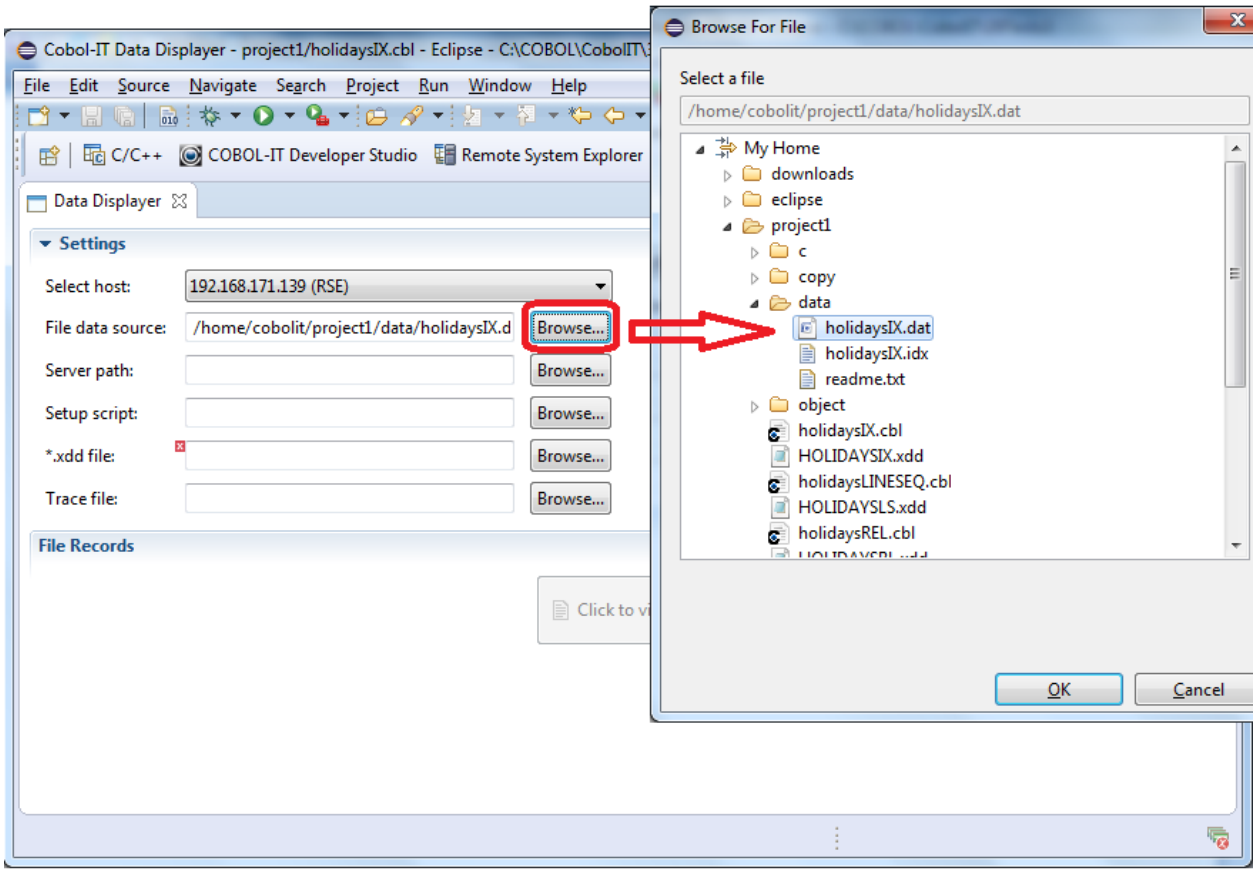
Dropdown the Select Host combo box, to see available local and remote connections. In our example, we will select our remote connection to a Linux Server. This remote connection has been established using the Remote System Explorer, and is connected during these tests. All source files, .xdd files, and data files in these tests are located on this remote Linux Server, as is the COBOL-IT Compiler Suite version 3.9.

(Early beta versions of the Data Displayer were not yet fully ported to Windows, so the Localhost option was not available.)



File data source

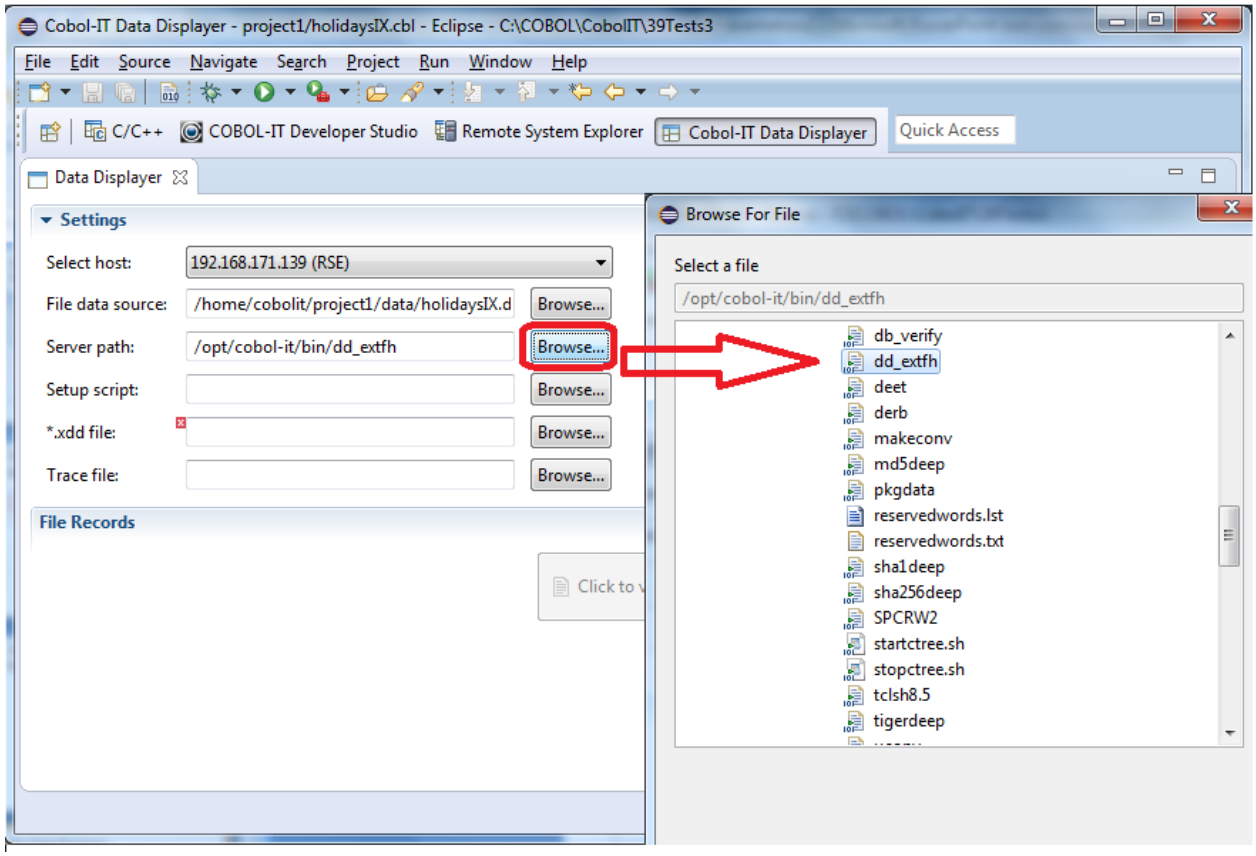
Click on the Browse button, and locate the data file you wish to display in the Data Displayer. From the Browse for File screen, select the file, and click OK to return to the Data Displayer.



Server path

Click on the Browse button, and locate the EXTFH server. The EXTFH server is named dd_extfh and is located in the bin directory of your COBOL-IT installation.

From the Browse for File screen, select the EXTFH server and click OK to return to the Data Displayer.



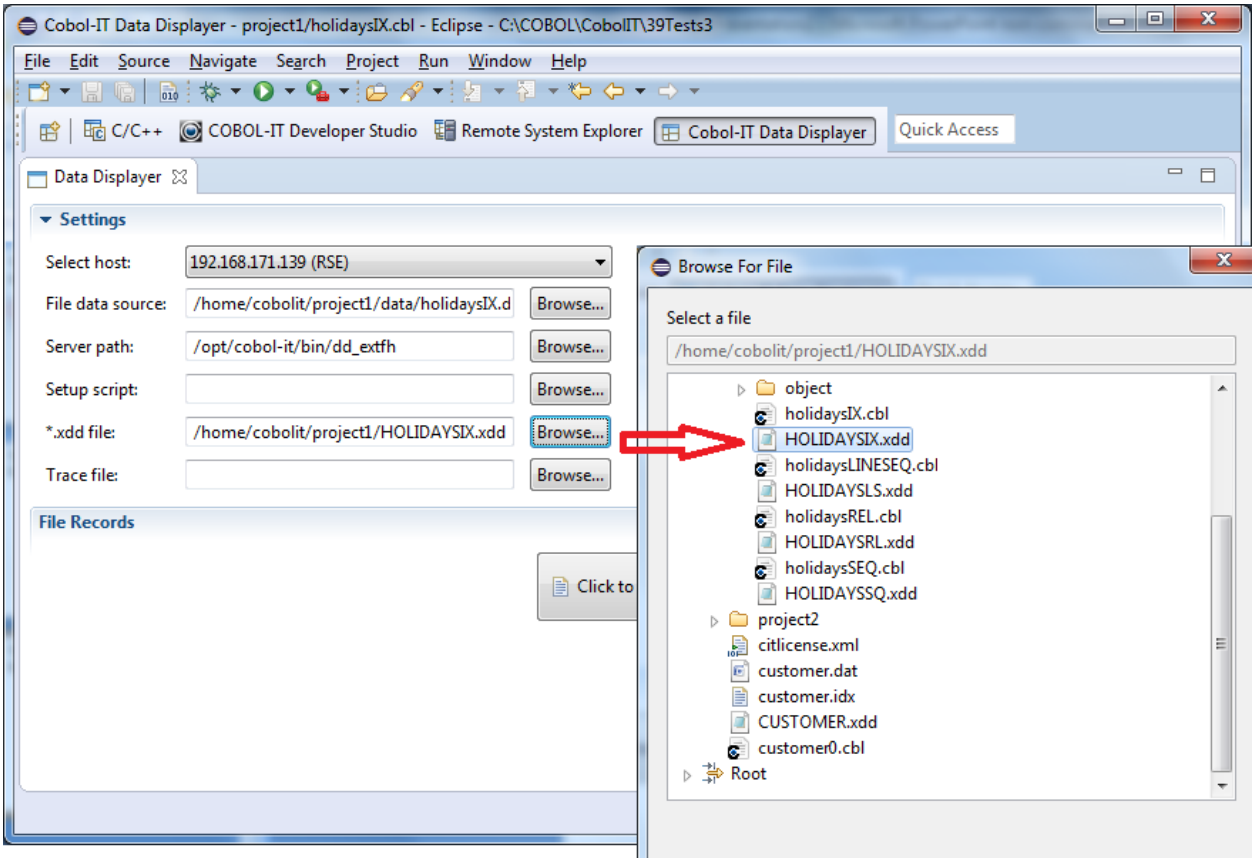
***.xdd file**

Selecting the *.xdd file

Click on the Browse button, and locate the XDD file for your File Data Source.

XDD files are derived from the FD of the file for the File Data Source. They are produced by the COBOL-IT compiler, by compiling a program that includes this FD with the –fgen-xdd compiler flag.

Our XDD file is called HolidaysIX.xdd. From the Browse for File screen, select the XDD file and click OK to return to the Data Displayer.



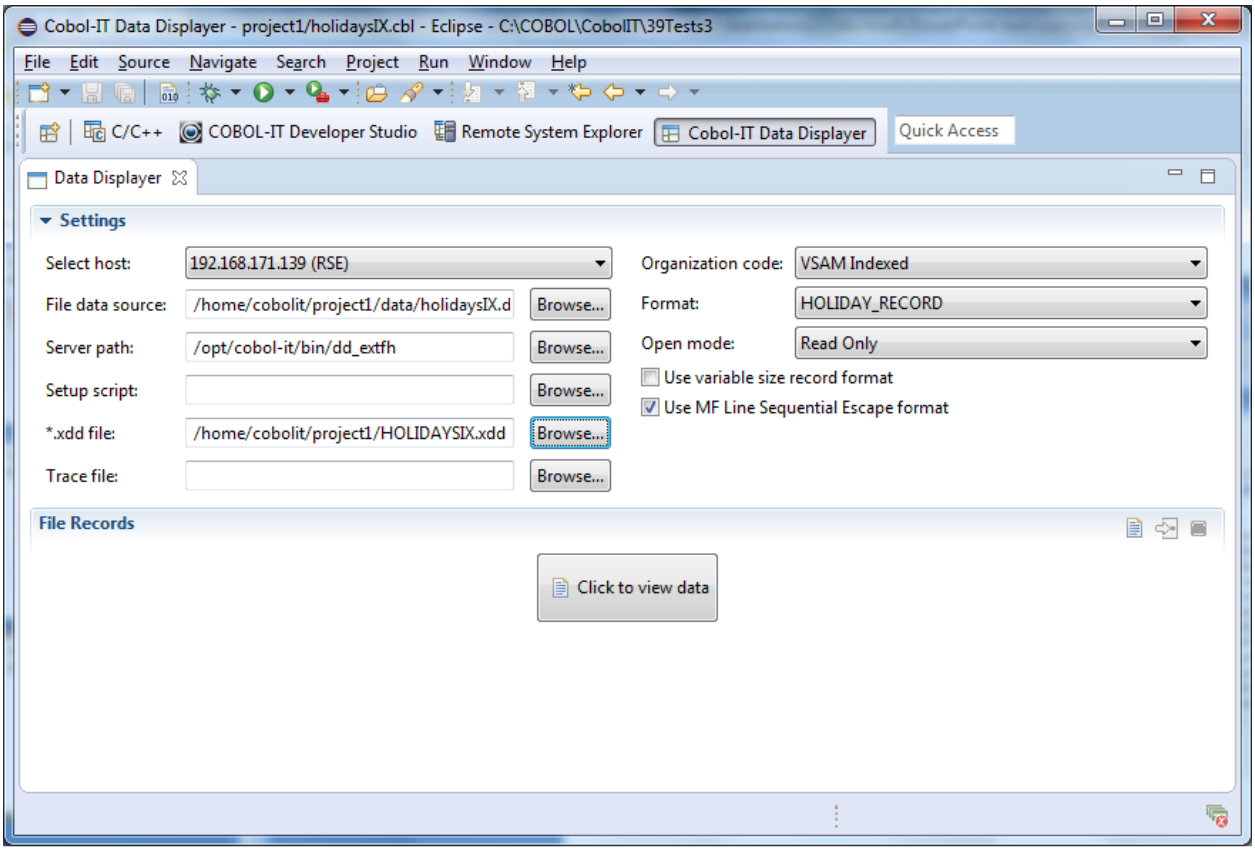
Information in the *.xdd file

This is the XDD file that we have selected.
When the XDD file has been selected, the Data Displayer screen is updated, based on the information contained within this file. This XDD is derived from a VSAM indexed file, with two record formats, which are named HOLIDAY_RECORD and HOLIDAY_RECORD_2.
The records are fixed length, and 70 bytes in length. You will notice when we open file for data display that the field names listed here will be transferred into the column headers. The two record formats have different field names, and these will be reproduced in the column headers of the file display.
We will see how you can toggle between record formats when we advance to the Data Display of this file.

```
<?xml version="1.0" encoding="US-ASCII"?>
<table name="HOLIDAYSIX" maxRecLen="70" minRecLen="70" >
<key duplicate="false" primary="true" >
<part name="HOLIDAY_NAME" offset="0" size="25" />
</key>
<select organization="2" lsmf="1" varrec="0" >
indexed
</select>
<schema name="HOLIDAY_RECORD" size="70" >
<field name="HOLIDAY_NAME" offset="0" size="25" type="Alphanum" />
<field name="HOLIDAY_DATE_WEEK_DAY" offset="25" size="9" type="Alphanum" />
<field name="HOLIDAY_DATE_THE_MONTH" offset="34" size="9" type="Alphanum" />
<field name="HOLIDAY_DATE_THE_DAY" offset="43" size="2" type="Alphanum" />
<field name="HOLIDAY_DATE_THE_YEAR" offset="45" size="4" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD" offset="49" size="8" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS" offset="57" size="8" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET" offset="65" size="5" type="Alphanum" />
</schema>
<schema name="HOLIDAY_RECORD_2" size="70" >
<field name="HOLIDAY_NAME_2" offset="0" size="25" type="Alphanum" />
<field name="HOLIDAY_DATE_2" offset="25" size="24" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_2" offset="49" size="21" type="Alphanum" />
</schema>
</table>
```

.xdd file updates the interface

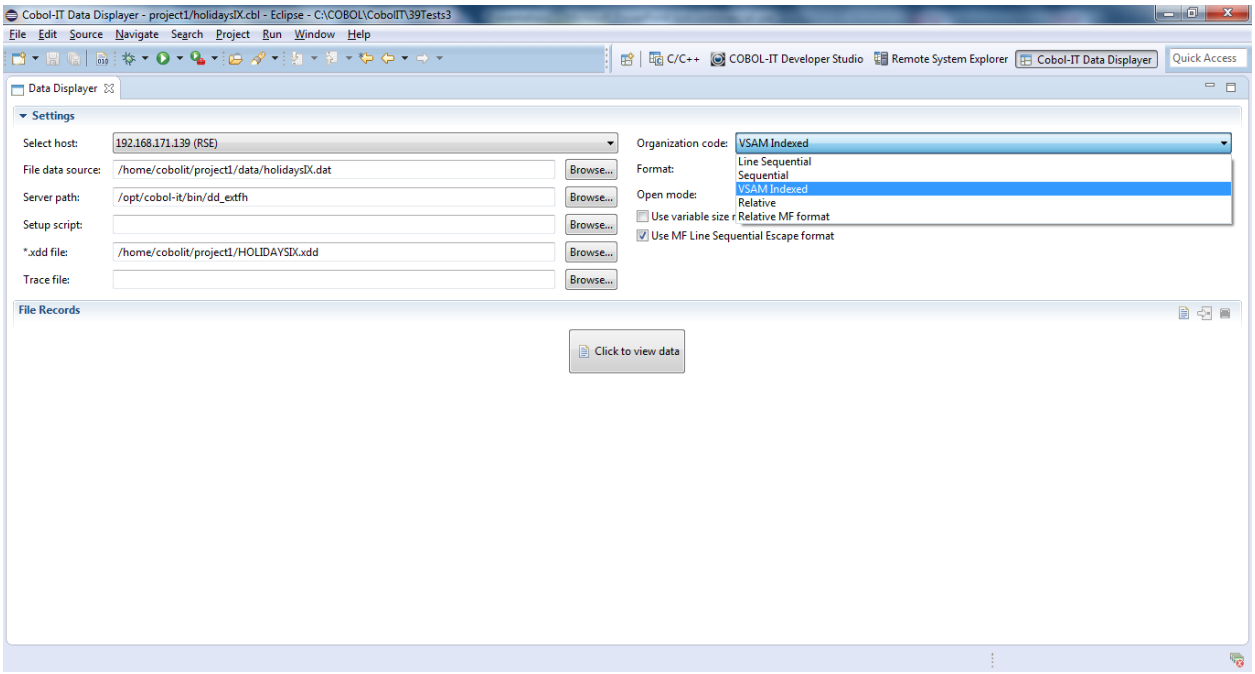
After the XDD file has been selected, you will see fields on the right-hand side of the screen, which are pre-filled from information within the XDD file.
The Organization Code is set to VSAM Indexed. This information has been extracted from the XDD file.
The Format is set to HOLIDAY-RECORD. This is the record name of the file. The Data Displayer has the ability to support multiple record formats. Our file has multiple record formats, HOLIDAY_RECORD and HOLIDAY_RECORD_2, which are viewable from the dropdown combo box. The user can toggle between the views of these two formats by toggling between these formats on the Format combo-box.
The Open Mode is set to Read Only. This is the Default.



Organization code

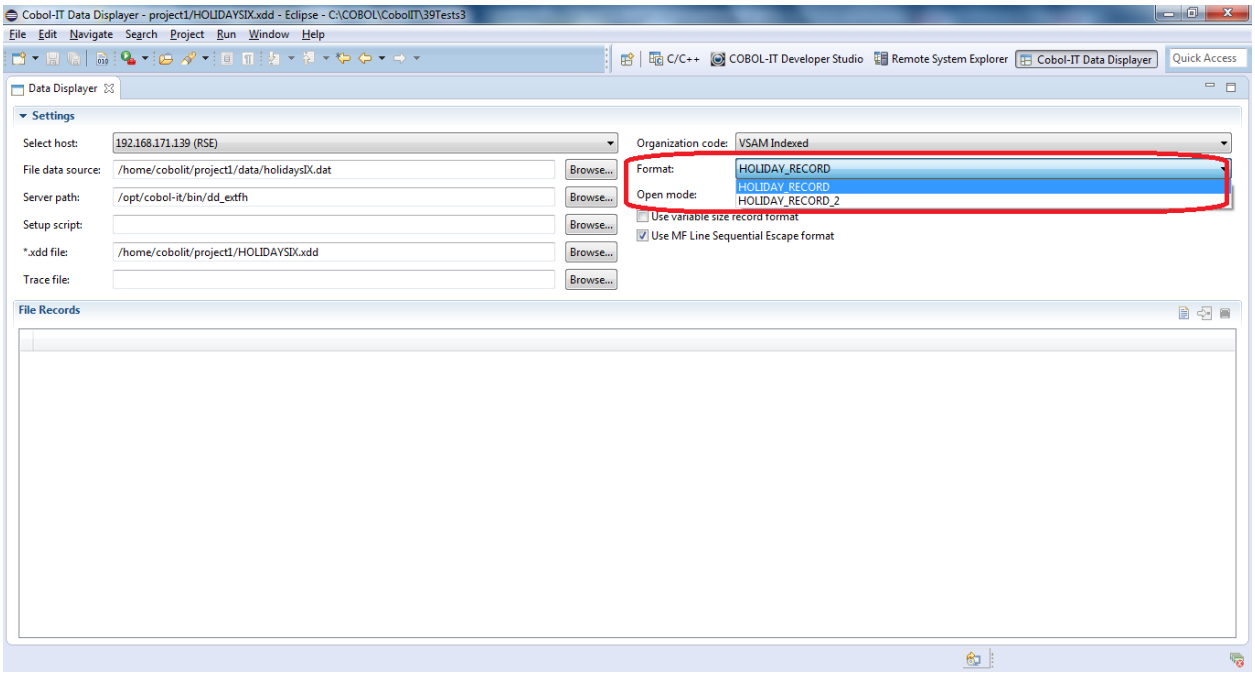
The Data Displayer can be used to display any of these types of files in the Organization drop-down combo-box.

Changing the selection has no effect.



Format

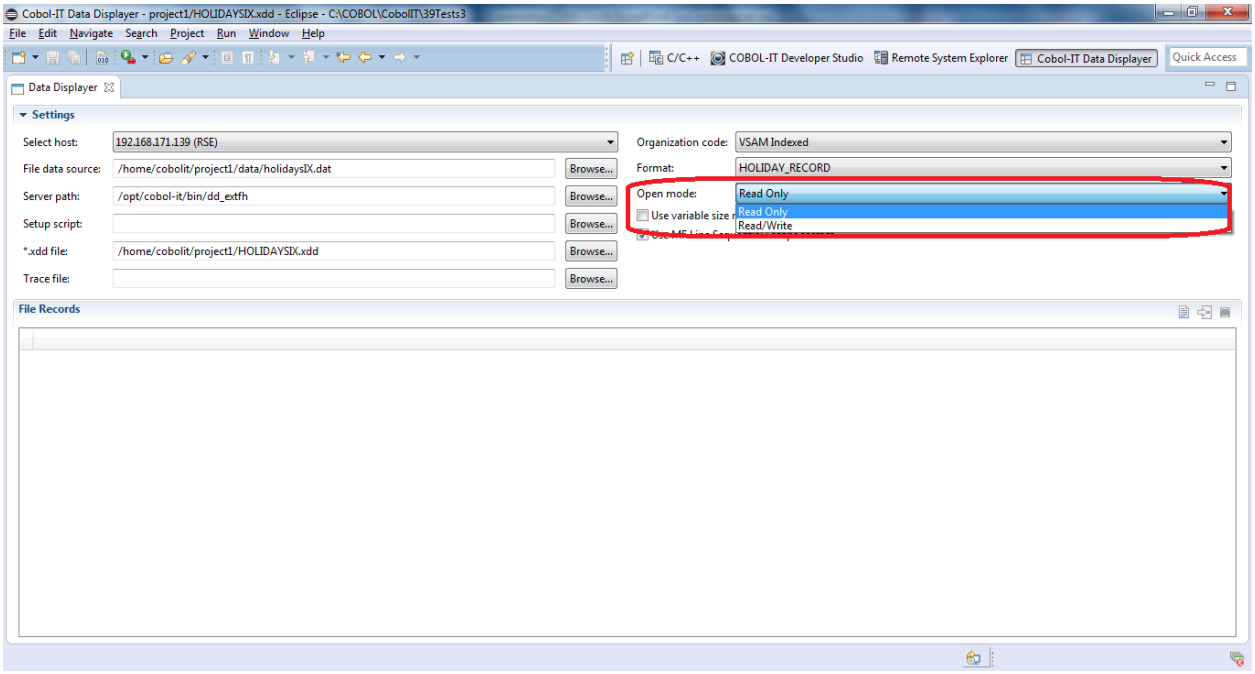
Our XDD has two record formats, and this is reflected in the choices available in the Format dropdown box. When a file is open in the Data Displayer, you can toggle between available data formats without closing and re-opening the file.



Open mode

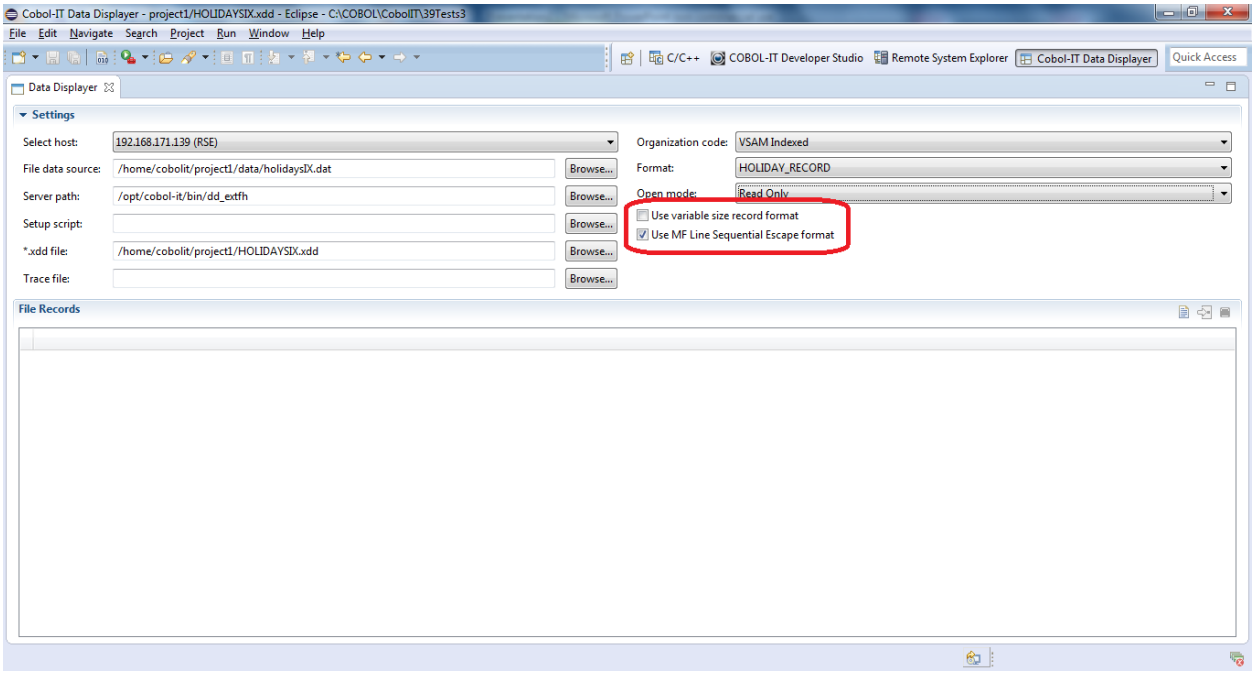
You can either open a file READ-Only (the default), or Read-Write.

You must select the Open Mode before opening the file in the Data Displayer. If you wish to change the Open Mode, you must close the file, and re-open it, using the toolbar buttons on the file Records display.



Variable/Line Sequential Formats

Variable and Line Sequential Formats are critical when handling different types of data files. The Data Displayer recognizes variable size record formats.
When using Micro Focus Line Sequential files, the Data Displayer can be set to match the MF Line Sequential Escape format.



The File Records Window

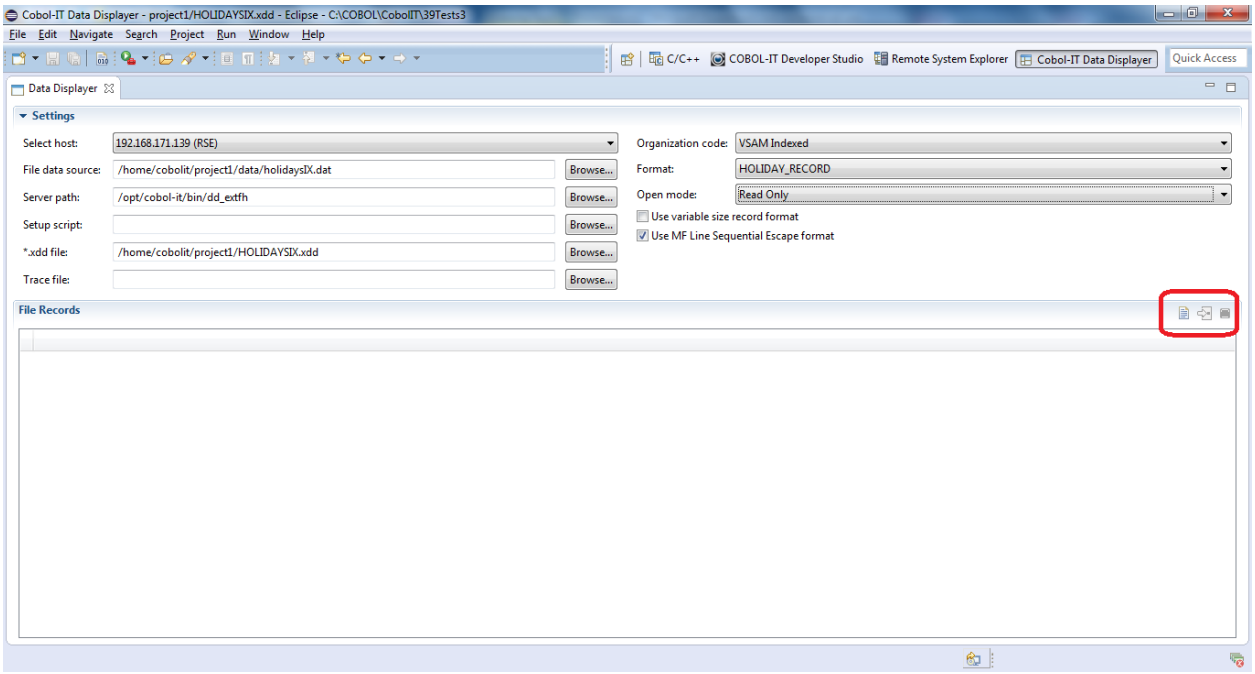
The File Records toolbar

The File Records Toolbar contains 3 buttons, which are (from left to right), the Open toolbar button, the “Go To” toolbar button, and the “Close” toolbar button.

Click on the Open button to open the data file, and populate the File Records window with data.

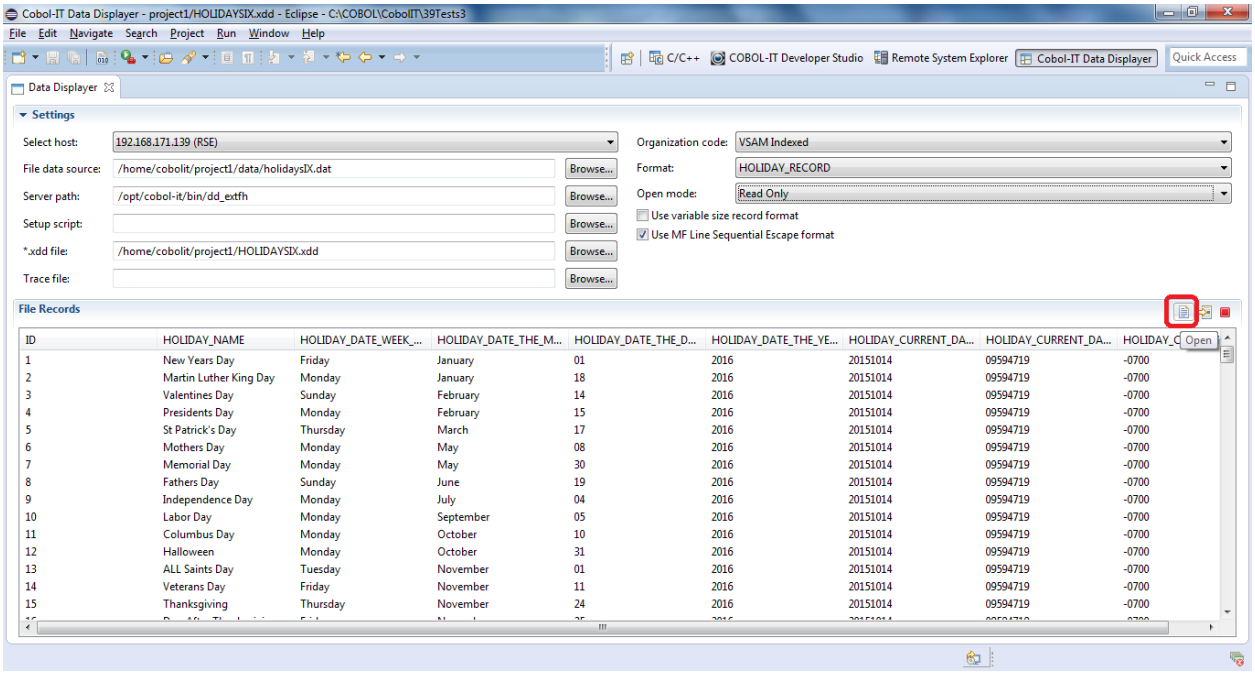
When you have Open’ed a file, Click on the “Go To” button to go to a line number in the file.

Click on the Close button to close the file, and clear the screen.



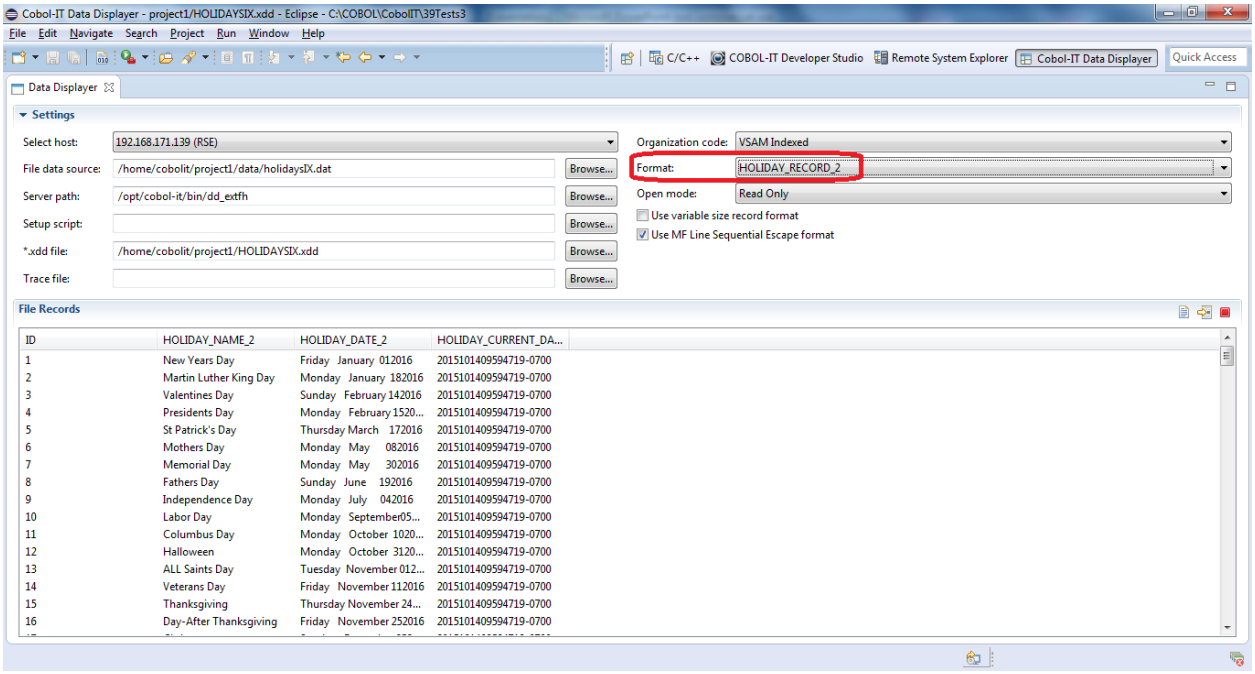
Display Records (Read-Only)

Click on the Open button on the File Records toolbar to open the file and display the file records. This file is open for Read Only, so you can scroll through the file, but cannot modify data.



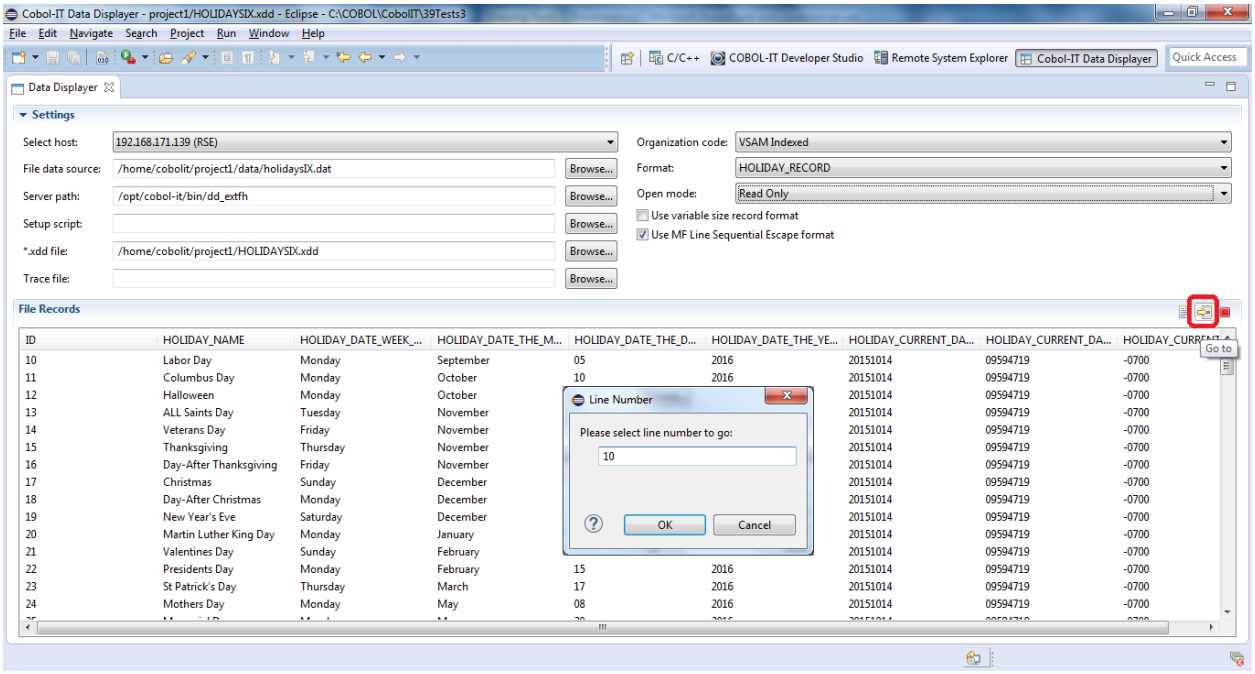
Toggle Record Format

Holiday-record-2 has fewer fields than Holiday-record. This is reflected in the fewer number of column headers. The data displayed is the same.



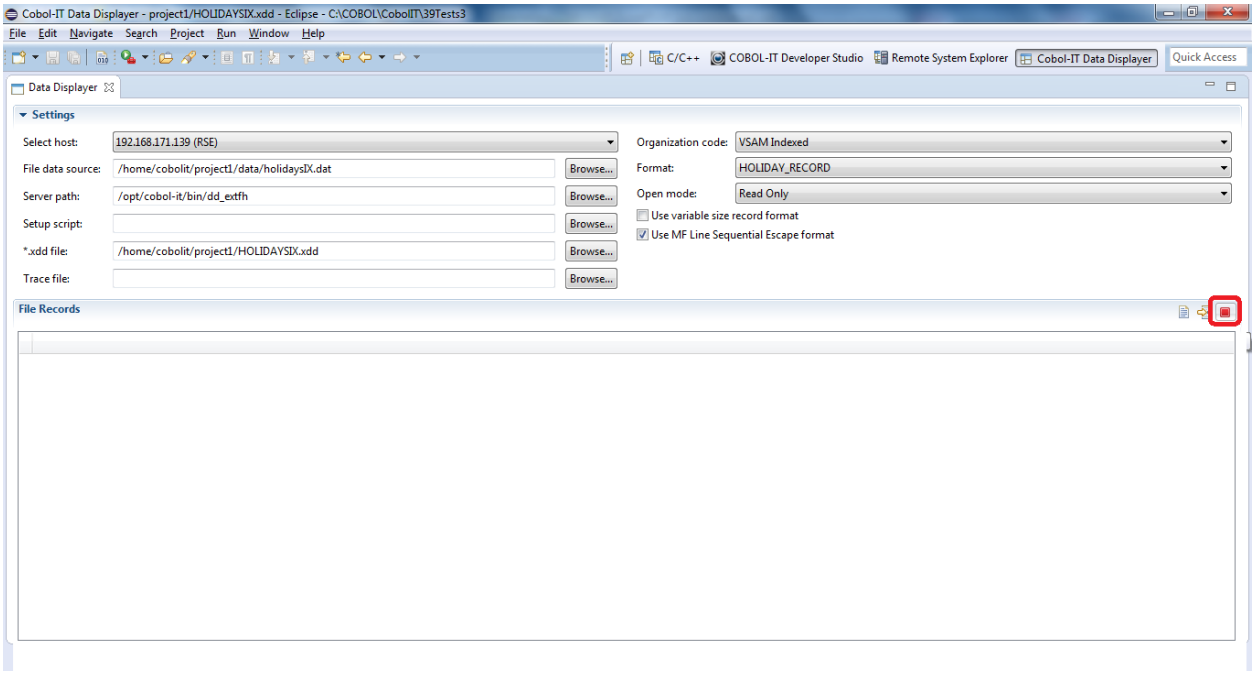
Go To Line Number

Click on the Go To button on the File Records Toolbar.
In the Line Number Window, enter a line number. In this exercise, we have entered line number 10.
The Data Display adjusts so that line 10 is at the top of the screen.



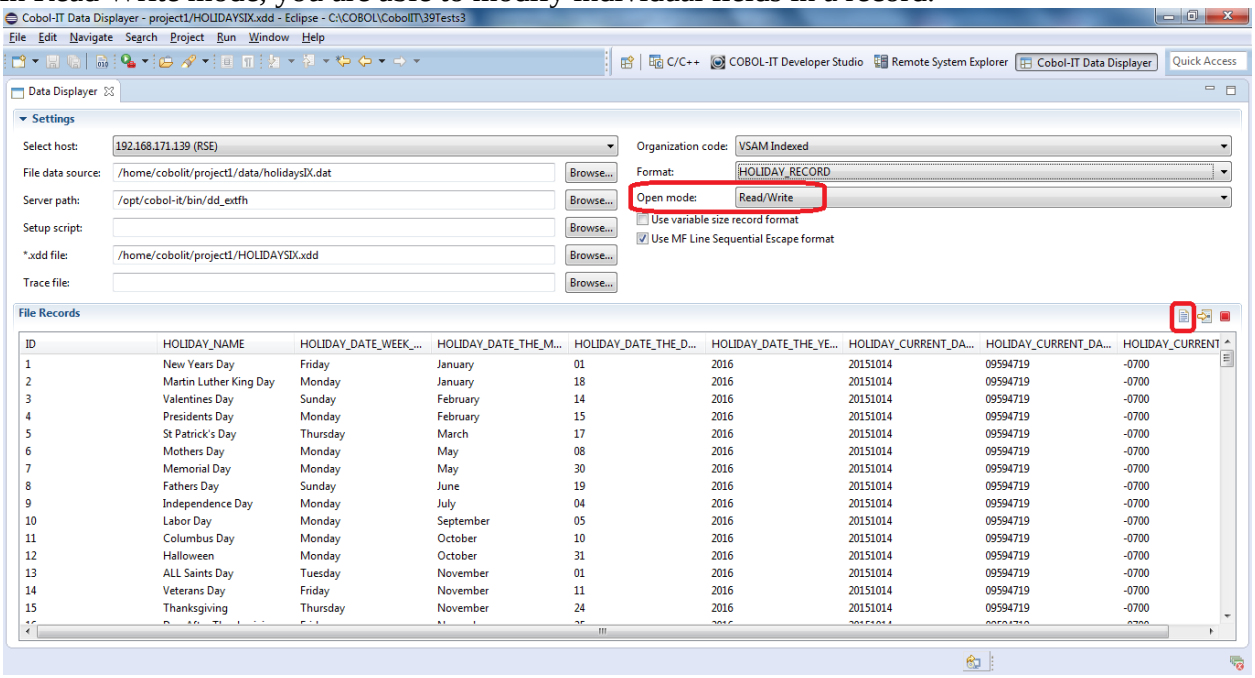
Close File

Click on the Close button on the File Records Toolbar to close the file and Clear the Screen. After closing the file, you can open Read-Write, and examine some more functionality.



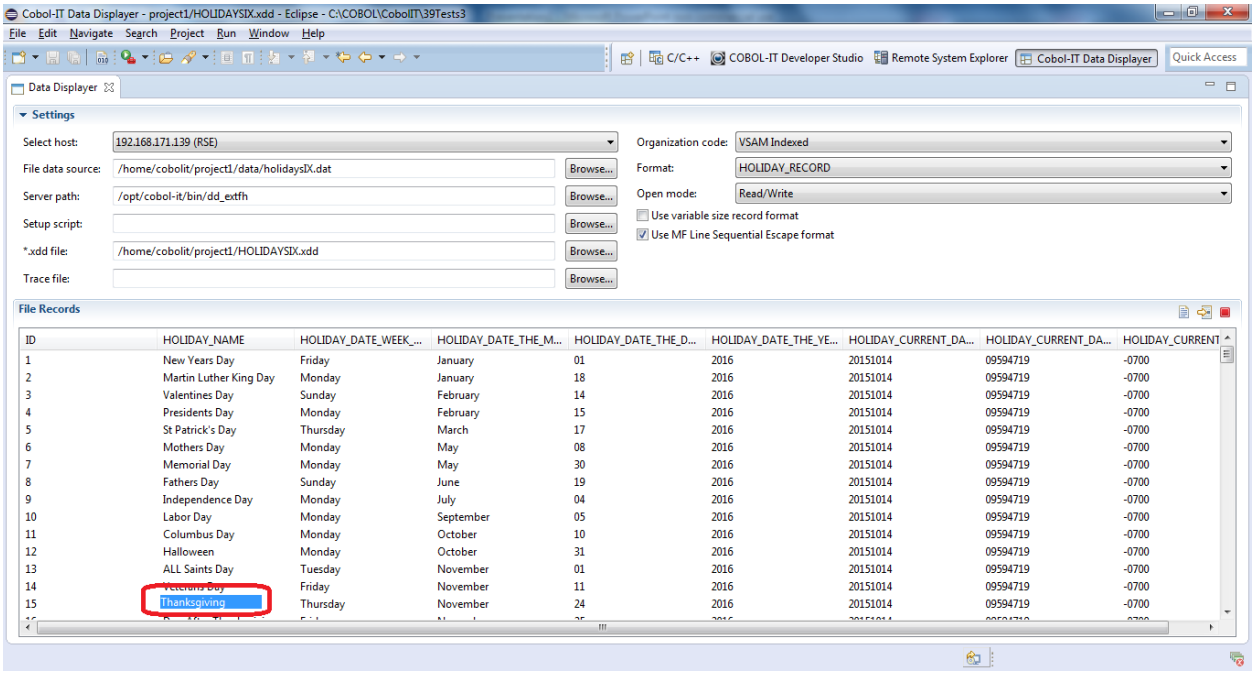
Display File Records (Read/Write)

To Open a File Read/Write, select Read/Write from the Open mode dripdown combo box. Click on the Open button on the File Records toolbar. This will open the file in Read Write mode. In Read Write mode, you are able to modify individual fields in a record.

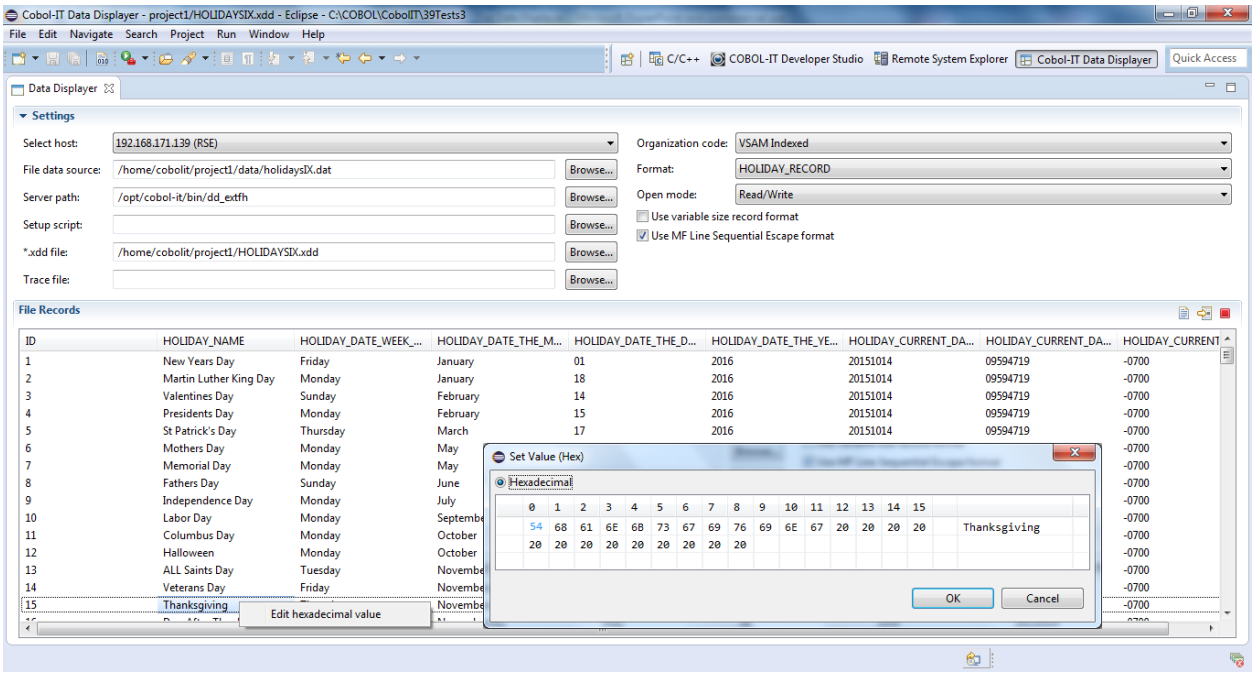


Modify a Data Element

When a file is open Read-Write, double-click on a data element in a record. The selected record is highlighted with a blue background. Single-click in this cell to activate the cursor with the cell displaying the data element. With your cursor active within the cell, you can modify the contents of the cell, typing from your keyboard.



Modify a Data Element (Hex)
When a file is open Read-Write, single-click on a data element in a record. The selected record is highlighted with a light-blue background. Right-click on this cell to see the “Edit hexadecimal value” button.
Select the “Edit hexadecimal value” button to see the “Set Value (Hex) window.
In the Set Value (Hex) window, enter any cell, and modify the hexadecimal value by overtyping the two-digit number with a new two-digit number.



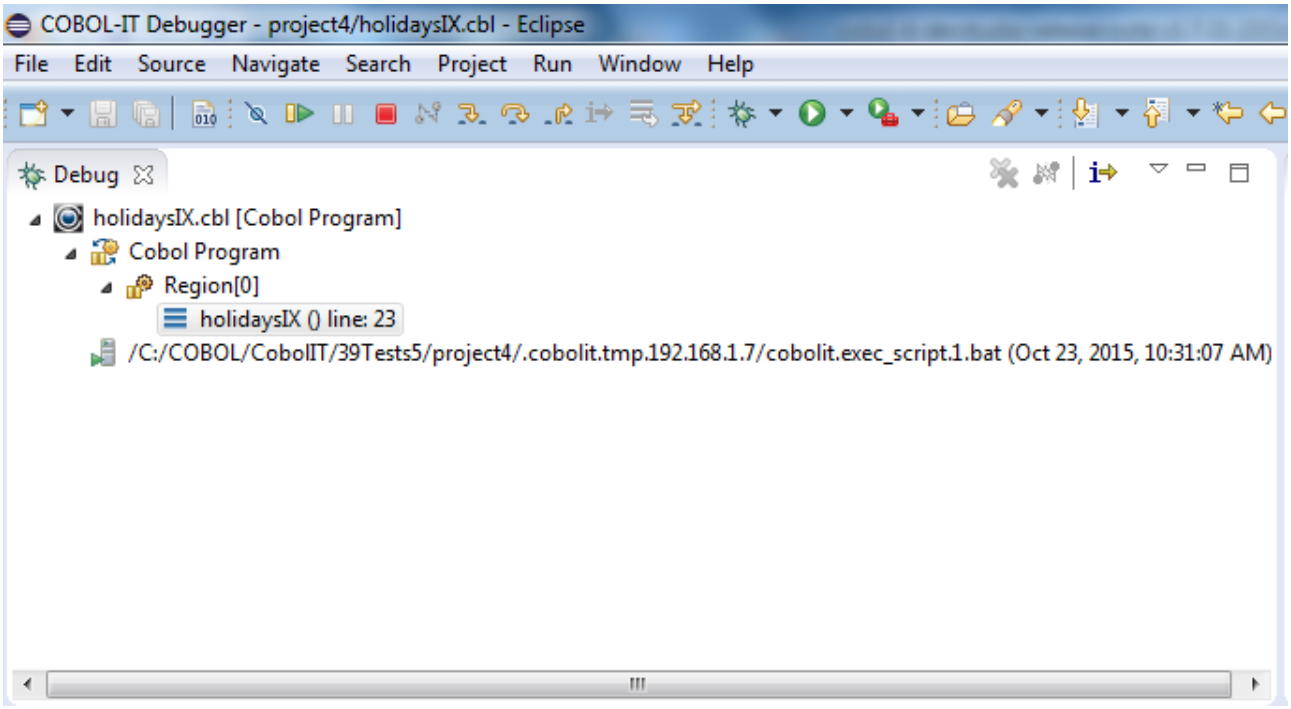
1.7.19 -fgen-xdd compiler flag

The-fgen-xdd compiler flag has been added to the “Files” tab in the COBOL Properties Window.

1.7.19 Debugging at the region level

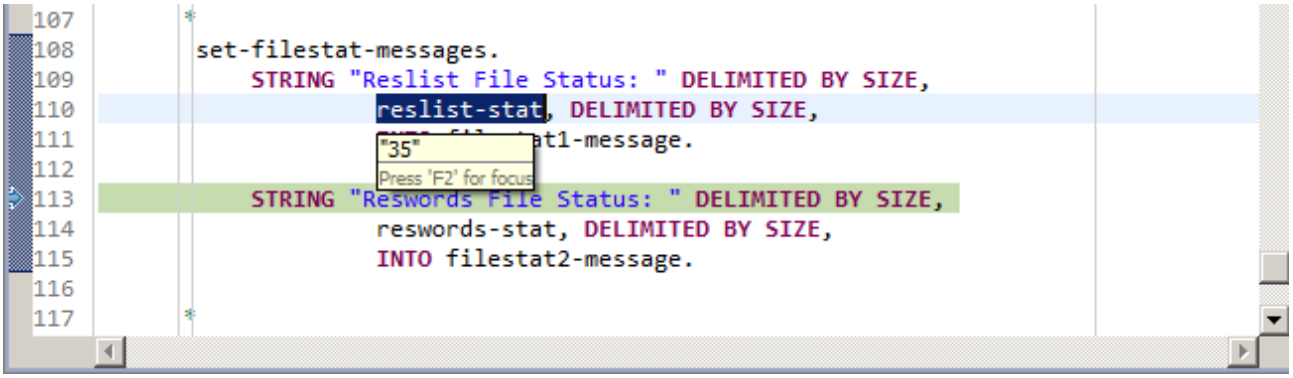
The Developer Studio automatically shows all regions used by the program. There is no need to re-connect when the program starts a new region. (Requires COBOL-IT Compiler version 3.9 and later).

The Debug View, displaying Region(0).



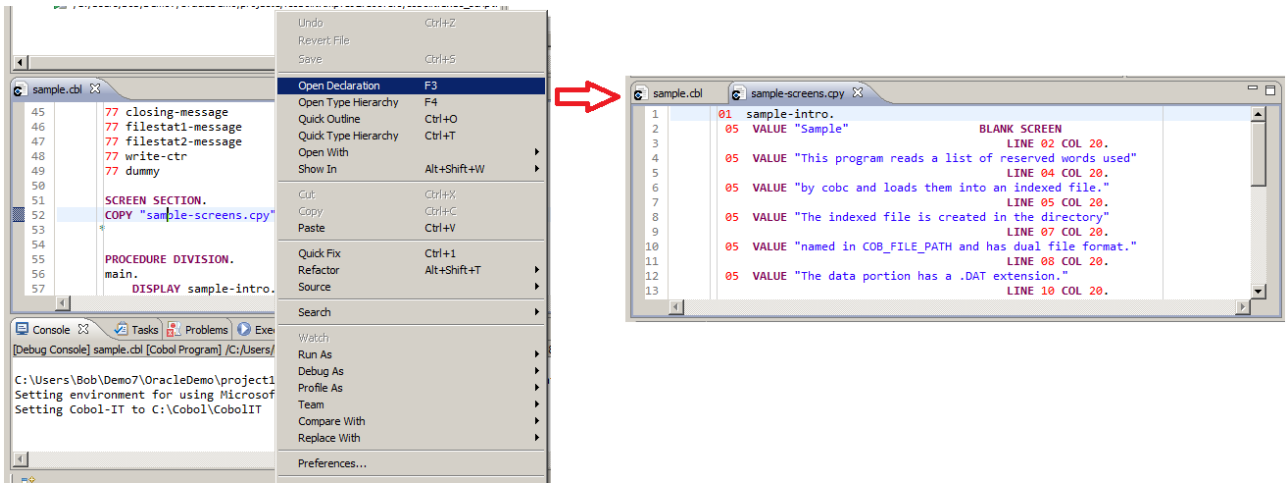
1.7.15 Hover over field displays value of field

When the cursor is hovered over a data field in the source code, the debugger displays the value of the variable.



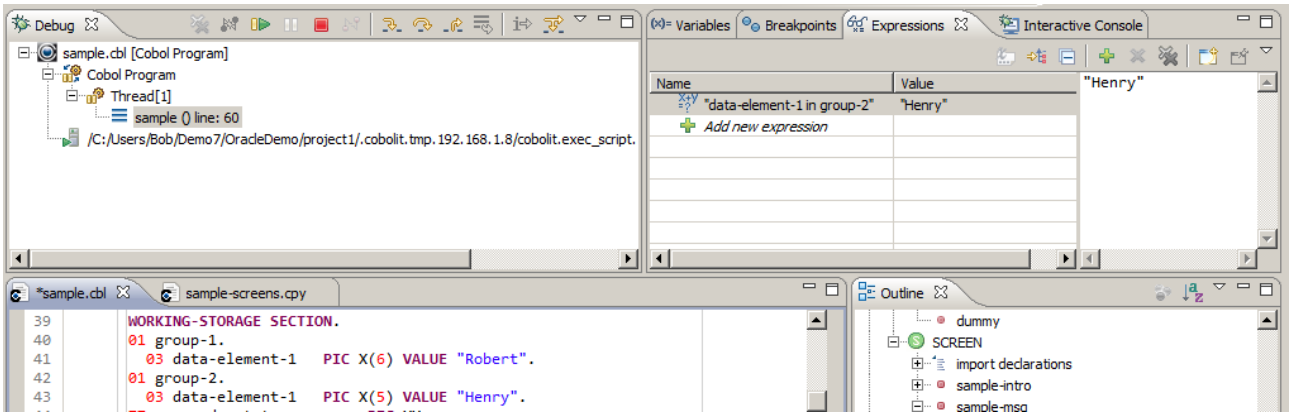
1.7.15 Press F3 on selected copyfile name opens Window containing the copyfile

Select copyfile and select the Open Declaration function from the right-click dropdown menu, or press the F3 function key to open the selected ccopyfile in a separate window in the Code Editor window.


sample.cbl
77 closing-message
77 filestat1-message
77 filestat2-message
77 write-ctr
77 dummy
SCREEN SECTION.
COPY "sample-screens.cpy"
PROCEDURE DIVISION.
main.
DISPLAY sample-intro.
Undo Ctrl+Z
Revert File
Save Ctrl+S
Open Declaration F3
Open Type Hierarchy F4
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Open With
Show In Alt+Shift+W
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Quick Fix Ctrl+1
Refactor Alt+Shift+T
Source
Search
Watch
Run As
Debug As
Profile As
Team
Compare With
Replace With
Preferences...
Console
Tasks
Problems
[Debug Console] sample.cbl [Cobol Program]
C:\Users\Bob\Demo7\OracleDemo\project1
Setting environment for using Microsof
Setting Cobol-IT to C:\Cobol\CobolIT
sample.cbl
sample-screens.cpy
01 sample-intro.
05 VALUE "Sample" BLANK SCREEN
LINE 02 COL 20.
05 VALUE "This program reads a list of reserved words used"
LINE 04 COL 20.
05 VALUE "by cobc and loads them into an indexed file."
LINE 05 COL 20.
05 VALUE "The indexed file is created in the directory"
LINE 07 COL 20.
05 VALUE "named in COB_FILE_PATH and has dual file format."
LINE 08 COL 20.
05 VALUE "The data portion has a .DAT extension."
LINE 10 COL 20.

1.7.15 Expression View in Debugger supports "FieldA in/of GroupB ..."

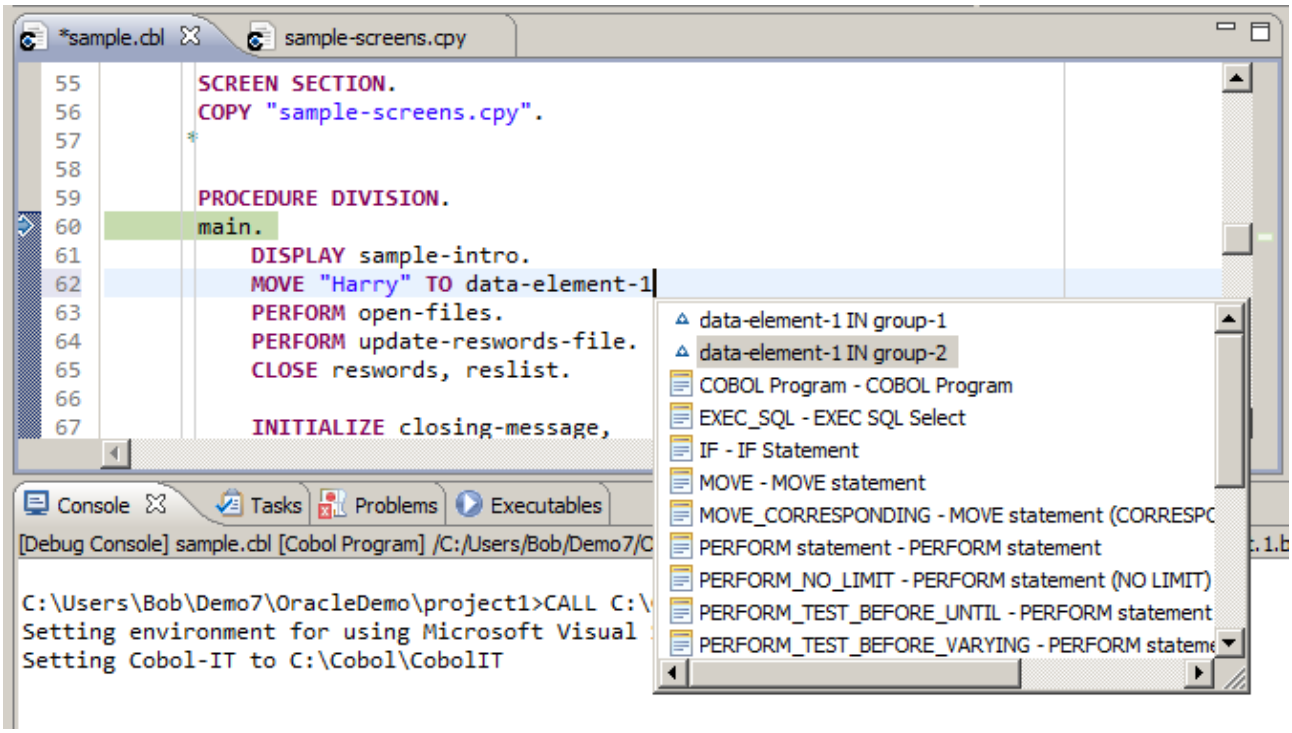
The debugger supports qualified names in the Expression View.



1.7.14 Copy files are parsed and included in the Outline

1.7.14 Field auto completion (CTRL-Space) is improved

The auto-completion function enables fully qualified data names, such as "`fielda in groupb`".



1.7.14 Field auto completion (CTRL-Space) is improved

The auto-completion function recognizes fields defined in copybooks.

1.7.14 Extensions to Window>Preferences>COBOL>Code Assist

The Code Assist options now include:

[ ] Always auto-complete full field path

When selected, causes the selected field to include full path information.

[ ] Use 'OF' instead of 'IN' for nested data

When selected, causes the auto-complete function for a qualified data item to be written with OF instead of IN. As an example, “data-element-1 OF group-2.”

Enhanced Support for Debug Attach

Note- Enhanced Support for Debug Attach requires Versions 3.7 or greater of the Compiler Suite EE.

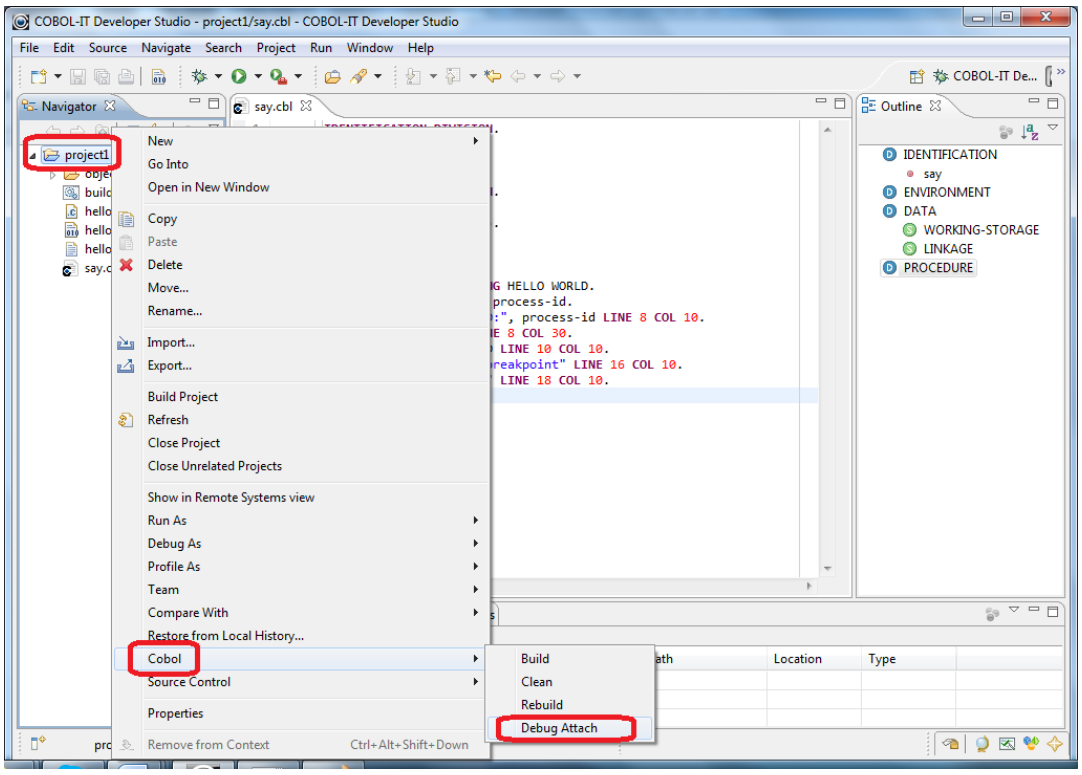
The COBOL-IT Developer Studio now provides enhanced support for the Debug Attach function, which allows the COBOL-IT Debugger to attach to a currently running process that has not been started in the debugger, which is identified either by its process id, or by its debug id.

The Enhanced Support for Debug Attach is very useful when running COBOL-IT programs in an Enterprise Solution Stack. In these environments, where COBOL is CALL’ed from an application such as a Transactional Monitor, which is not started in the Developer Studio, it is critical to be able to have a way to interrupt the process, and attach the debugger.

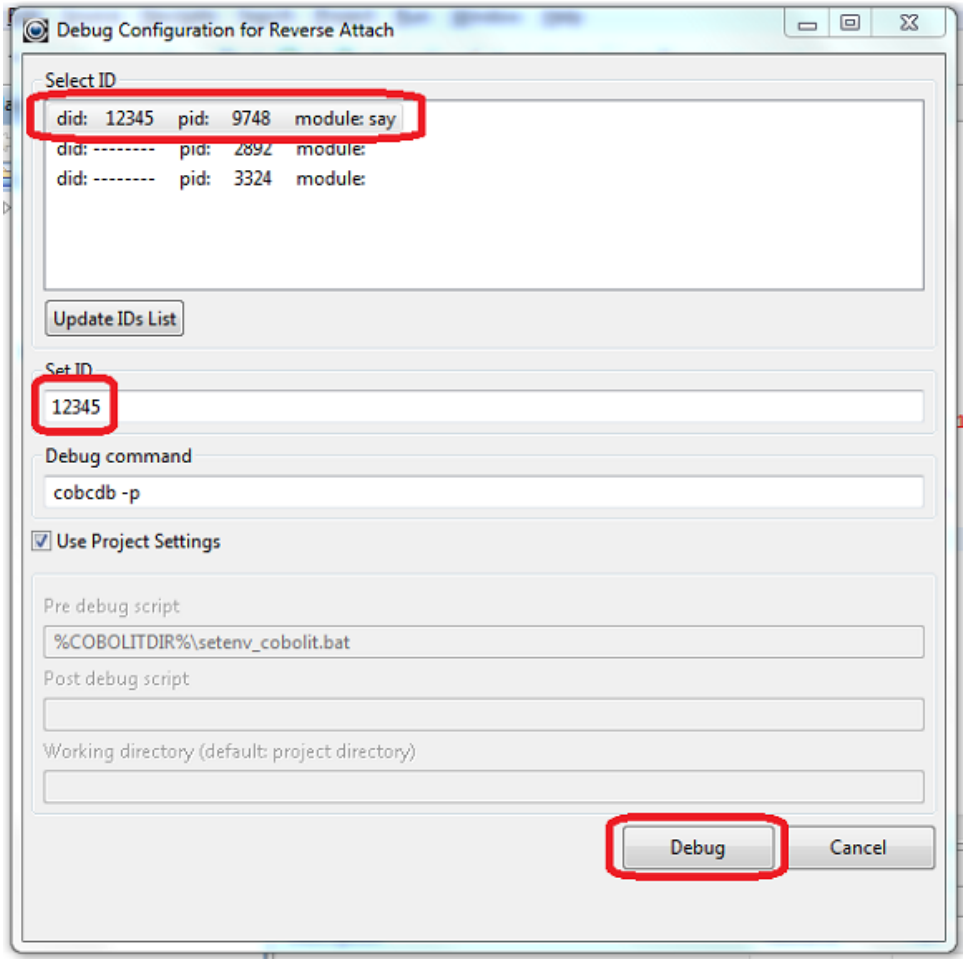
For the purpose of this example, we have launched an executable called “hello.exe”, as follows:

```
set COB_DEBUG_ID=12345
set COB_LIBRARY_PATH=.\object
cobc -x -flink-only -o hello hello-dynamic.c
hello
```

To enter the Debug Attach interface, select [project name] in the Navigator Window, and select the function COBOL>Debug Attach.

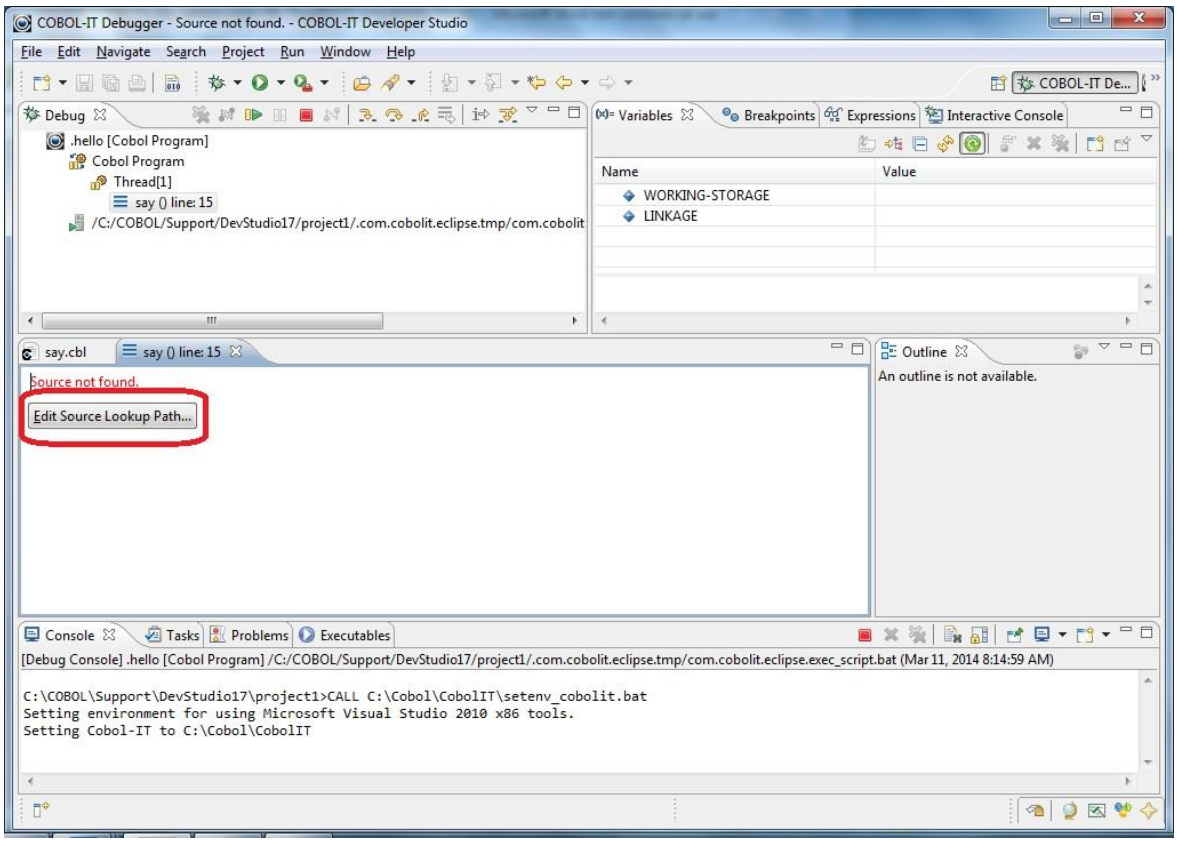


This opens the Debug Configuration for Reverse Attach Window:
Select the process from the Select ID interface. This will prefill the Set ID entryfield.
Click on the “Debug” button.

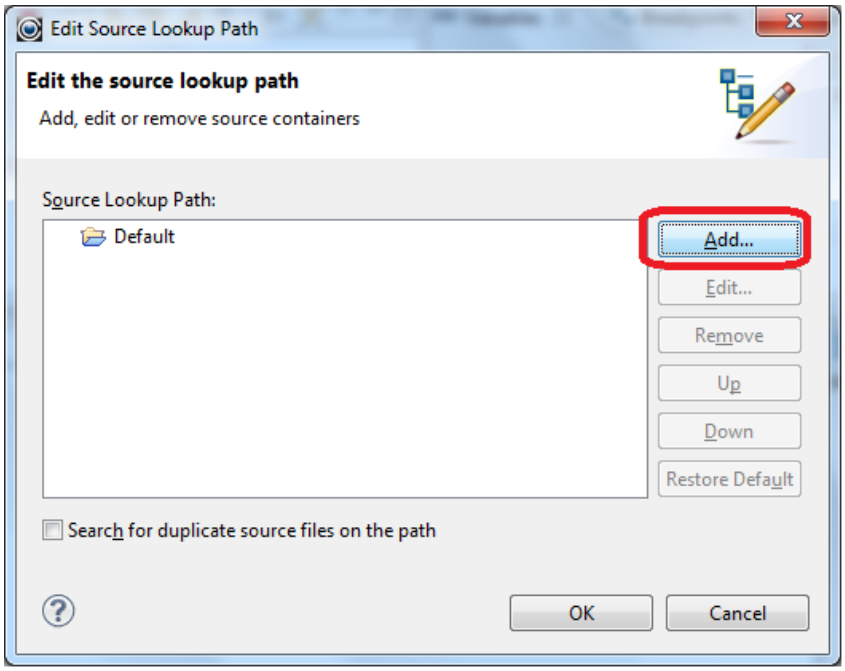


After clicking on the debug button, in our sample, hit « enter » to terminate the ACCEPT on whch the sample program is paused

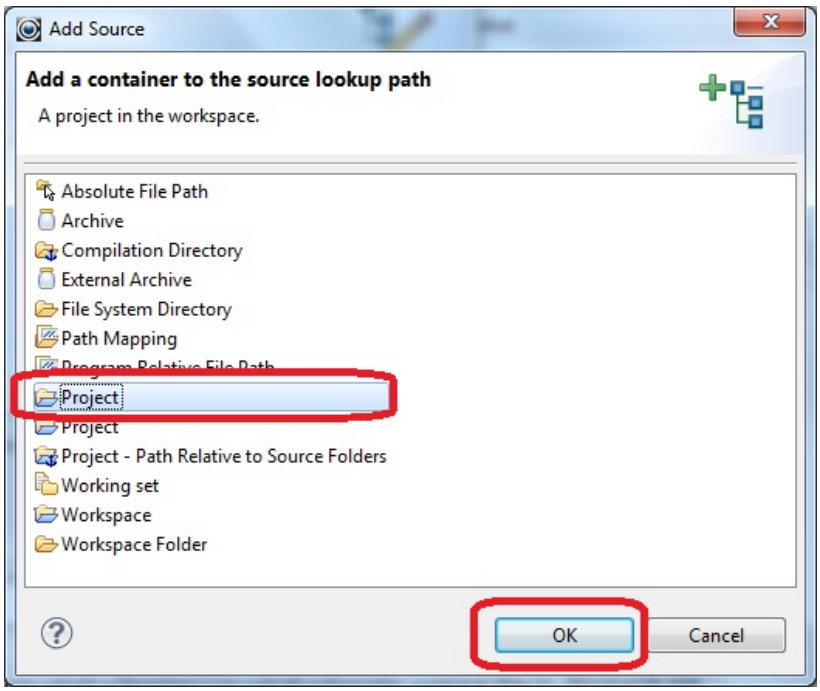
The Debugger will open, and pause on the first executable line of code. If this is the first time using the Debug Attach functionality in a Project, you will have to indicate where to provide a “Source Lookup Path”.



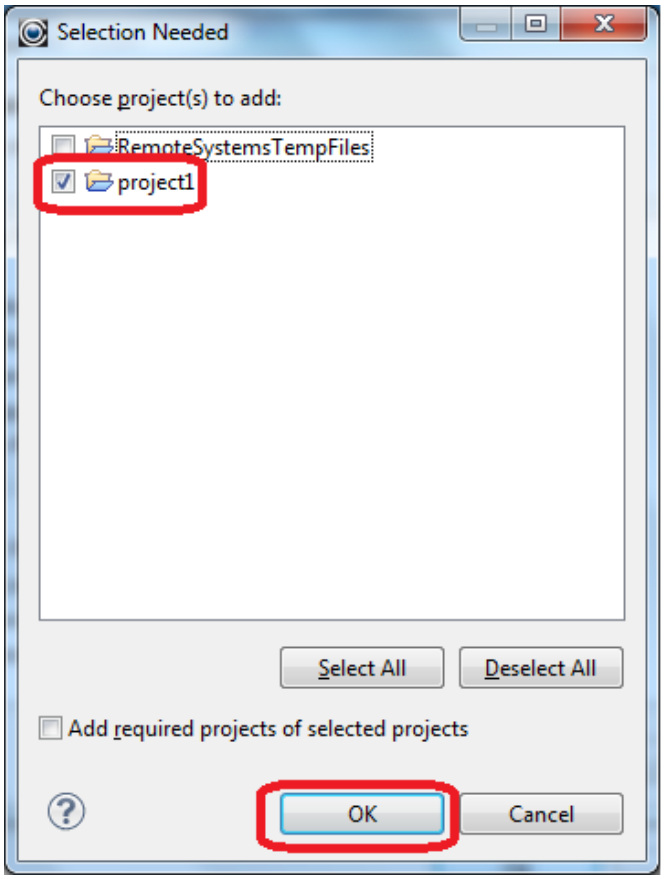
Select (Add) On the Edit Source Lookup Path Interface:



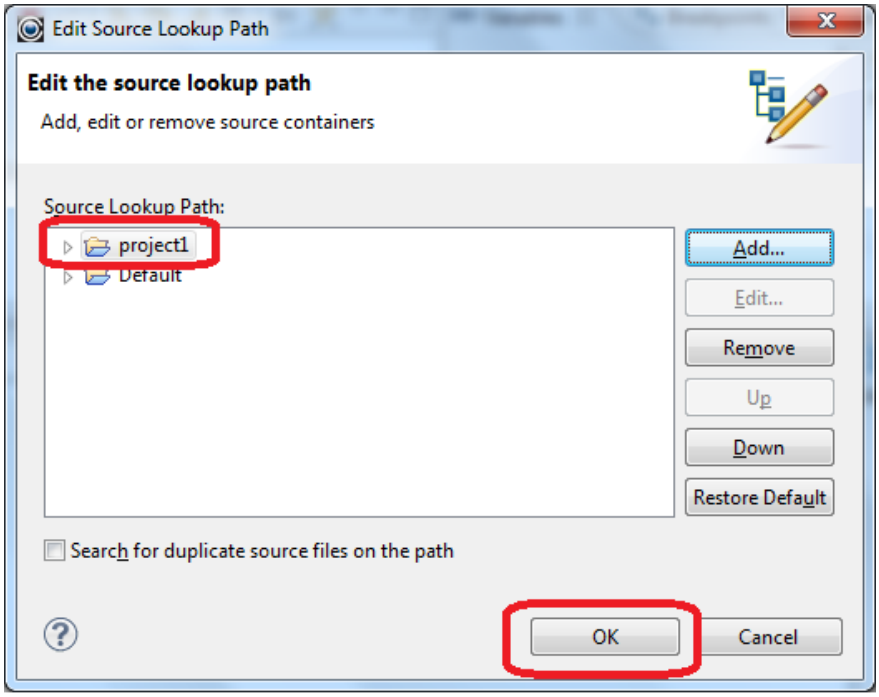
In our example, the source file is located in the project folder-



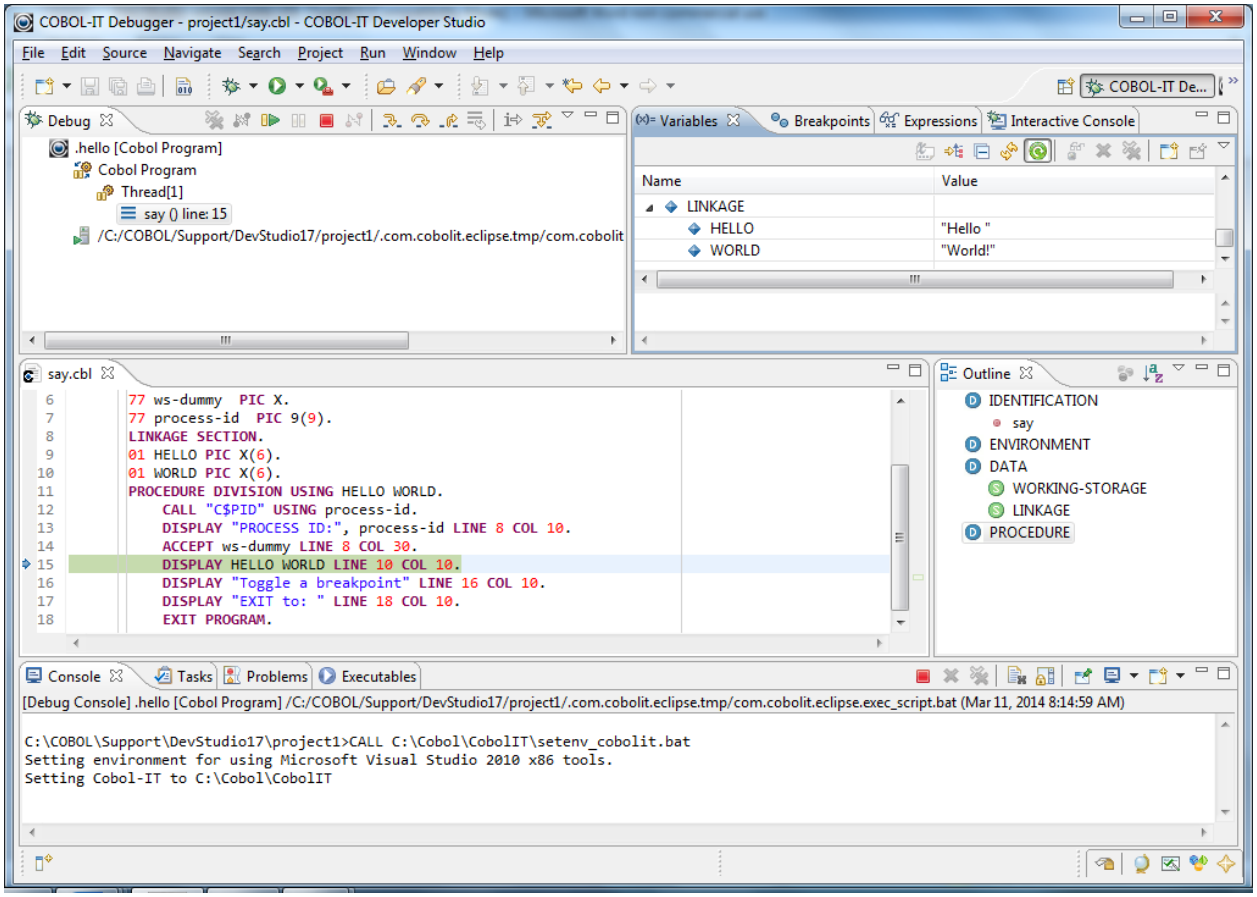
Select project1 from the dialog window, and click “Ok”.



Return to Edit Source Lookup Path screen, select project1, and click “Ok”.



You will now enter the Debugger with Source Code. Here, you can step, set breakpoints, use all of the functionality of the COBOL-IT debugger. Note that subsequent uses of the Debug Attach functionality will not require you to locate the source code, before entering the Debugger.



Note- for purposes of this example, we used the following source , in Windows:

Buildit.bat

```
set COB_DEBUG_ID=12345
set COB_LIBRARY_PATH=.\object
cobc -x -flink-only -o hello hello-dynamic.c
hello
```

hello-dynamic.c

```
/* hello-dynamic.c */
#include <libcob.h>
static int (*say)(char *hello, char *world);
int main()
{
/* COBOL-Runtime data */
/* COB_RTD is a macro that define rtd variable*/
COB_RTD = cob_get_rtd();
int ret;
char hello[7] = "Hello ";
char world[7] = "World!";
cob_init(rtd, 0, NULL);
/* find the module with PROGRAM-ID "say". */
say = cob_resolve(rtd, "say");
/* if there is no such module, show error and exit */
if (say == NULL) {
fprintf(stderr, "%s\n", cob_resolve_error (rtd));
exit(1);
}
/* call the module found and exit with the return code */
ret = say(hello, world);
printf("C program: %s\n", "hello-dynamic.c");
return ret;
}
```

say.cbl

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. say.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77 ws-dummy  PIC X.
       77 process-id  PIC 9(9).
       LINKAGE SECTION.
       01 HELLO PIC X(6).
       01 WORLD PIC X(6).
       PROCEDURE DIVISION USING HELLO WORLD.
           CALL "C$PID" USING process-id.
           DISPLAY "PROCESS ID:", process-id LINE 8 COL 10.
           ACCEPT ws-dummy LINE 8 COL 30.
           DISPLAY HELLO WORLD LINE 10 COL 10.
           DISPLAY "Toggle a breakpoint" LINE 16 COL 10.
           DISPLAY "EXIT to: " LINE 18 COL 10.
           EXIT PROGRAM.
```

Additions to COBOL Properties Interfaces

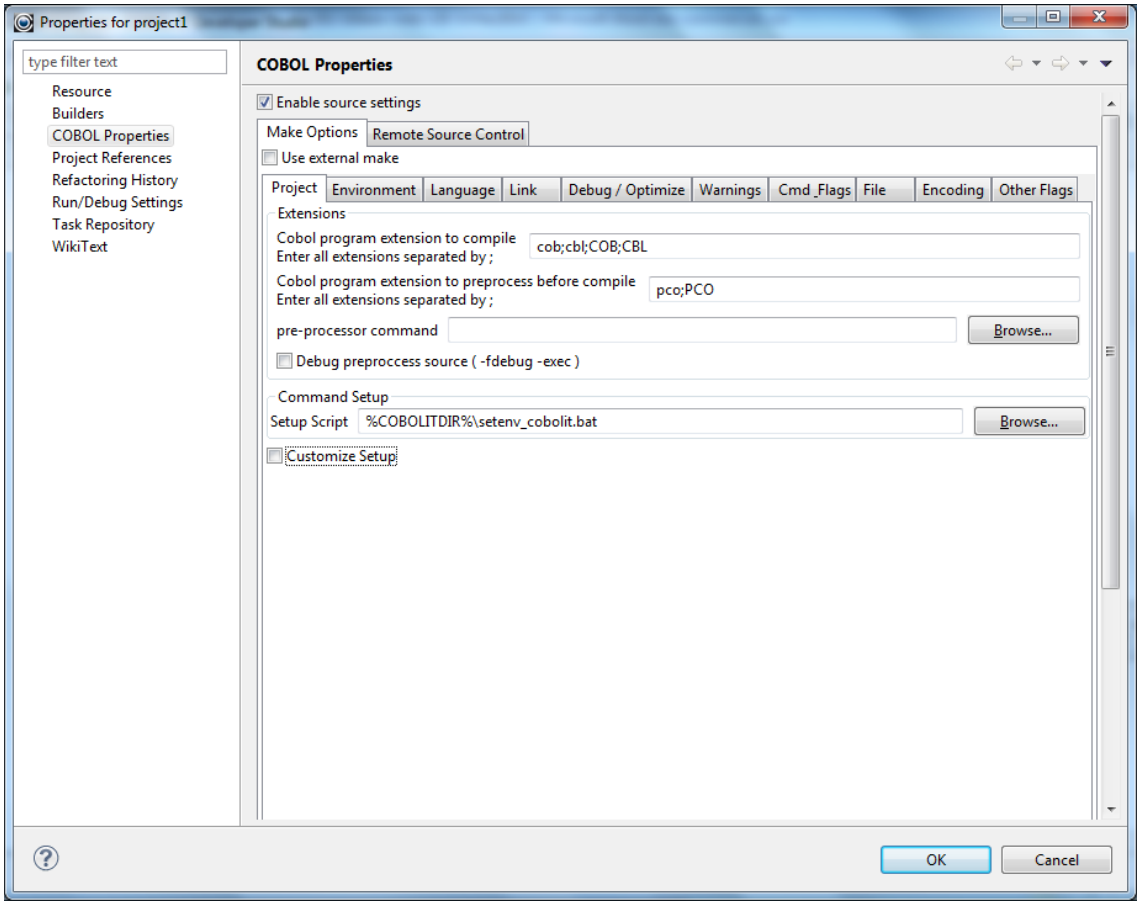
The COBOL Properties Interfaces, at different levels		
Workspace	Main Menubar	Window>Preferences>CO-BOL> Compiler
Project	Right-click dropdown menu on Project name	Project>Properties>COBOL Properties
Folder	Right-click dropdown menu on Source Folder name	Folder>Properties>COBOL Properties
Source File	Right-click dropdown menu on Source File name	File>Properties>COBOL Properties

As an overview,

- The COBOL Properties Interfaces have been enhanced to support all of the currently supported COBOL-IT Compiler flags.
- The Source/Project tab has been added, to organize your setup scripts, and scripts associated with use of the –preprocess compiler flag.
- File, Encoding, and "Other" tabs have been added to accommodate a number of compiler flags that were not formerly represented in the Developer Studio.

The COBOL Properties Window

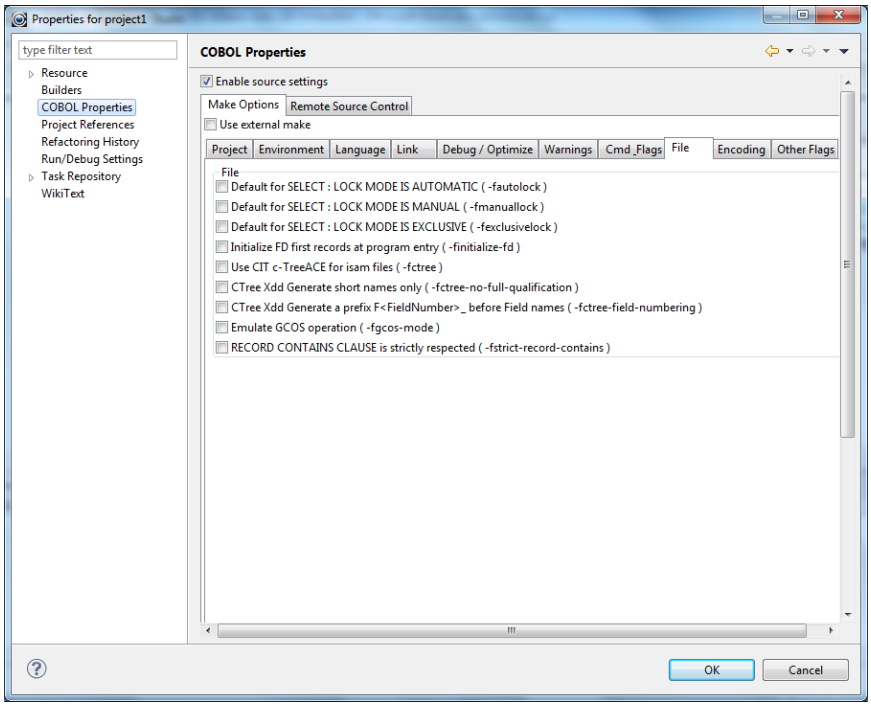
COBOL Properties>Project tab (or Source tab)



Prompt	Usage
COBOL program extension to compile	Programs with these file extensions in folders that have been added to the project, or in the project root directory will be compiled during the Build operation.
COBOL program extension to preprocess before compile	Programs with these file extensions in folders that have been added to the project, or in the project root directory will be pre-compiled with the –preprocess command defined during the Build operation.
Preprocessor command [Browse]	Browse, and select the batch file/script that is to be used with the –preprocess command. Entering a batch file/script file name here has the effect of adding the compiler command -preprocess=[batch file/script file] to the compile command for files with extensions designated for pre-compiling.
Debug pre-process source (-fdebug-exec)	Adds the compiler flag –fdebug-exec to the compiler command line. By default, the debugger will step through pre-compileed source lines. If you wish to have the compiler step through original source, compile with –fdebug-exec.
Command Setup	You may either enter a single setup script (the default), or you may elect to Customize your setup.
Setup Script [Browse]	The default behavior would be to select the COBOL-IT setup script, for example, setenv_cobolit.bat, in Windows.
Customize Setup	Causes the default Setup Script entry to be disabled. The Customized Setup area allows you to name different scripts for the Pre-Clean, Post-Clean, Pre-Build, Post-Build, Pre-Run, Post Run, Pre-Debug, Post-Debug states.

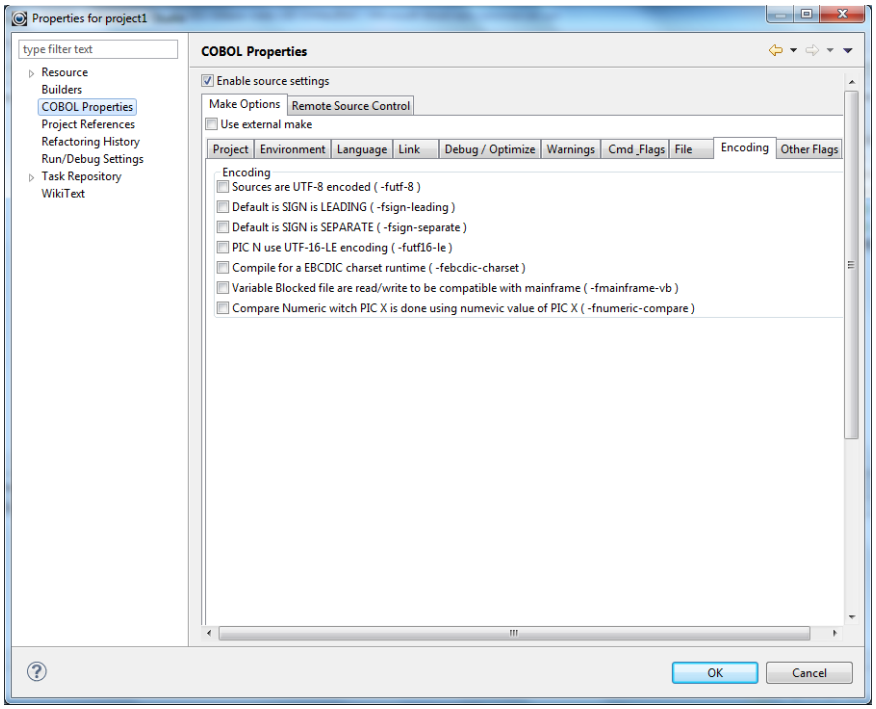
COBOL Properties>File tab

Includes compiler flags oriented towards file handling behaviors.



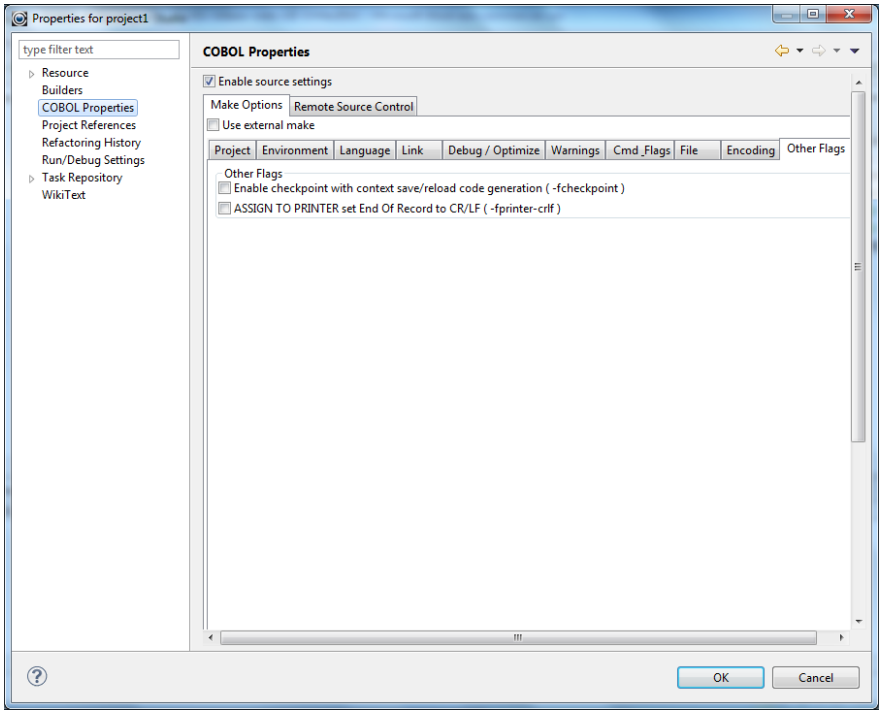
COBOL Properties>Encoding tab

Includes compiler flags oriented towards data encoding.



COBOL Properties>Other Flags tab

A catch-all area for flags that were not included on other tabs.



Runtime part of the debugger now run in a separate thread

With this enhancement, the Debugger GUI (Deet or Eclipse) continues to be available even when the process is active. While the process is running, the Debugger can read a field value, for example. Breakpoints can be set when the runtime process is paused on an ACCEPT statement.

Fixes

1.7.21 Fix expressions evaluation in debug mode.

ID: #1147036700

In previous versions (1.7.19. 1.7.20), the Expressions View in the Debugger Perspective failed to display expressions in some situations.

This has been corrected.

1.7.21 Fix the ability to modify a variable value in text or hex.

It is now possible to modify a variable value in either ASCII text or Hex.

1.7.21 Debugger aborts while pressing Step Over (F6) or Resume (F8) too quickly.

In some situations, the debugger could abort if the Step Over (F6) or Resume (F8) buttons were pressed too quickly.

This has been corrected.

1.7.21 Omit "FILE" prefixe while requesting variable values from debugger

ID: 1146941682e

If you hovered over a field-name from an FD in the source, you would see the field definition but no value.

This has been corrected.

1.7.21 Show correct definition while hovering over items in copybooks

ID: 1146941682f

If you hovered over a field in a copybook FD, you would see the description of last field in the FD.

This has been corrected.

1.7.21 Fix parser to correctly process string literals

ID: 1146941682a

When debugging, hovering over a field on the first line would not return the value of the field.

This has been corrected.

1.7.21 Fix parser to allow single letter variables

ID: 1146941682c

Hovering over a 1-character variable-name did not return a value. Also, the 1-character variable name was incorrectly listed in the Outline view as "PIC" instead of the 1-character name of the variable.

These anomalies have been corrected.

1.7.21 Fix copybooks opening by using F3/Open Declaration in an Editor View

ID: 1146944750

Copybooks could not be opened using F3/Open Declaration in an Editor view

This has been directed.

1.7.21 Edit hexadecimal value for expression in the format "FieldA in/of Group B"

ID: 1146944756

It was not possible to edit an expression in the format "Field-A in/of Group-B" in hex.

This has been corrected.

1.7.21 By default, don't show "import declarations at outline view

ID: 1146944774

When a copybook was displayed in the Outline view, it would include the label "import declarations". This could be confusing in some situations.

The "import declarations" label has been removed.

1.7.21 Fix auto-complete for copybooks in a project subdirectory specified by –I

ID: 1146944764

When you pressed Ctrl/space to auto-complete a field name, it would not check copybooks that are in a Project subdirectory (specified by –I).

This has been corrected.

1.7.21 Store COBOL launch configuration in the project it defined by default

The launch configuration for a program is now stored in a projectin the .launches folder. This allows you to examine the exact launch configuration used for a program, and save it for future use and examine it for diagnostic purposes. The launch configuration is stored in an xml file in the .launches folder in your project.

As an example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<launchConfiguration type="com.cobolit.eclipse.launching.CobolLaunchConfigurationType">
<stringAttribute key="CobolDebugCommand" value="cobcdb"/>
<stringAttribute key="CobolPostDebugScript" value=""/>
<stringAttribute key="CobolPostRunScript" value=""/>
<stringAttribute key="CobolPreDebugScript" value="%COBOLITDIR%\setenv_cobolit.bat"/>
<stringAttribute key="CobolPreRunScript" value="%COBOLITDIR%\setenv_cobolit.bat"/>
<stringAttribute key="CobolRunCommand" value="cobcrun"/>
<intAttribute key="CobolSSHTunnelingRemotePort" value="8484"/>
<intAttribute key="CobolSSHTunnelingRemotePortRange" value="50"/>
<booleanAttribute key="CobolUseProjectSettings" value="true"/>
<booleanAttribute key="CobolUseSSHTunneling" value="true"/>
<stringAttribute key="CobolWorkingDirectory" value=""/>
<stringAttribute key="mainScript" value="holidaysIX.cbl"/>
<stringAttribute key="nature" value="com.cobolit.eclipse.core.nature"/>
<listAttribute key="org.eclipse.debug.core.MAPPED_RESOURCE_PATHS">
<listEntry value="/project4/holidaysIX.cbl"/>
</listAttribute>
<listAttribute key="org.eclipse.debug.core.MAPPED_RESOURCE_TYPES">
<listEntry value="1"/>
</listAttribute>
<stringAttribute key="process_factory_id" value="org.eclipse.dltk.launching.scriptPro-
cessFactory"/>
<stringAttribute key="project" value="project4"/>
</launchConfiguration>
```

1.7.21 Add ability to evaluate several expressions at once.

This speeds up the evaluate when many expressions are present.

1.7.15 Unbound buildpath error when project opened in IDE

ID : 1146371432

The Developer Studio could return an Unbound Buildpath Container error when opening a Project. This report did not interfere with any of the normal operations of the Developer Studio, but was an unexpected behavior.

If you encounter this behavior, then the easiest way to correct it is to re-create the project in a new directory. The COBOL-IT technical team does not currently have advice on how to correct the problem from within the existing project.

1.7.15 Debugger aborts when using a table of PIC 1 USAGE BIT data items

ID: 1146336600

The usage of a table of PIC 1 USAGE BIT data items could cause the debugger to crash in some situations.

This has been corrected.

1.7.15 Developer Studio debugger display anomaly

In some situations, the Developer Studio debugger could incorrectly display a literal.

This has been corrected. The UTF-8 protocol now used by the Developer Studio debugger allows for the correct display of PIC X (non-national) fields.

1.6

1.6.0

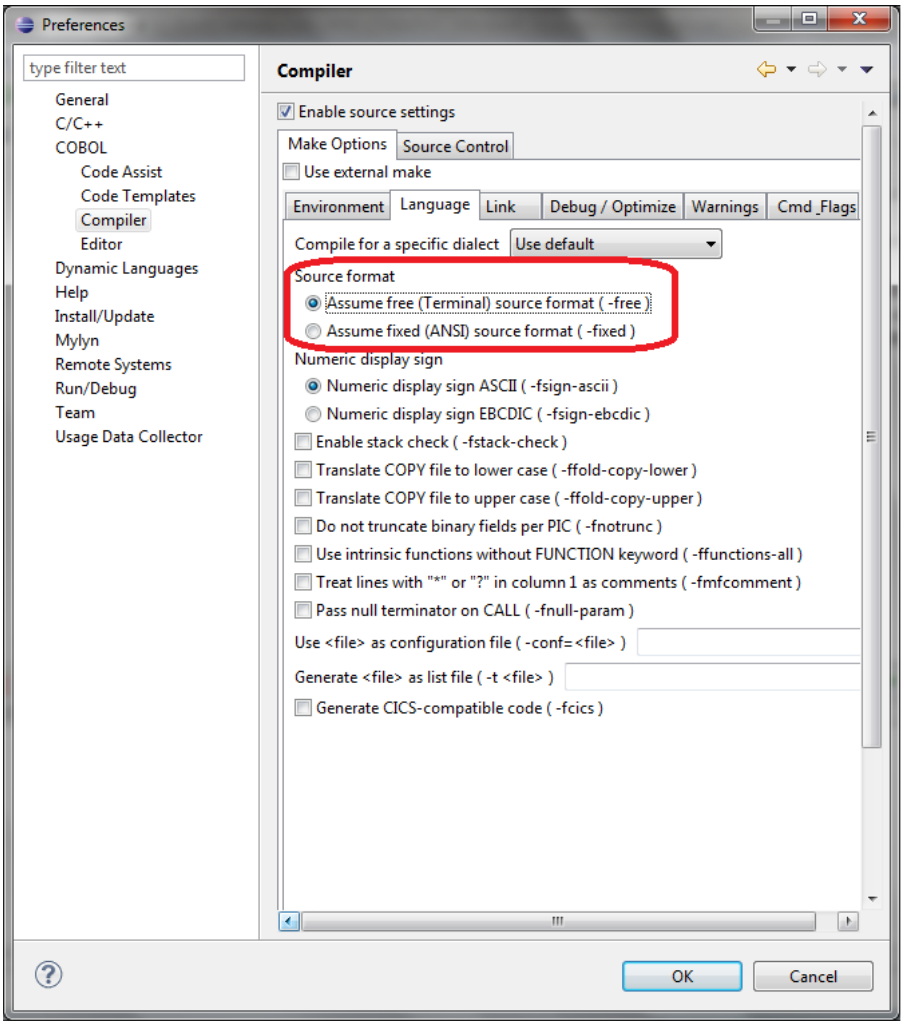
New

Support full for Free format Source

The COBOL-IT Developer Studio Code Editor now supports source files in terminal source format.
To use source files in terminal source format in the COBOL-IT Developer Studio, set the –free compiler flag. The –free compiler flag can be set at the Workspace, Project, or File level.

To set compiler flags at the Workspace level, in Window>Preferences>COBOL>Compiler>Language>
Select Source format>Assume free (Terminal) source format (-free)

When set at the Workspace level, the setting is the default for all Projects in the Workspace. When set at the Project level, the setting becomes the default for all files in that Project. When set at the file level, the setting applies only to the single file for which it is set.



Source files in terminal format have the following characteristics:

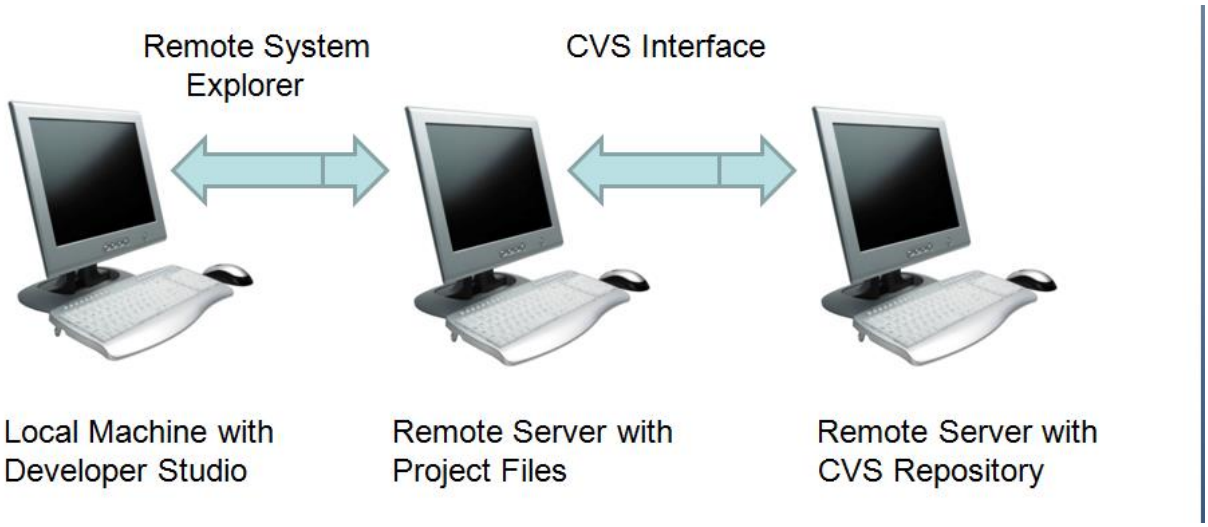
For a comparison of the characteristics of Terminal, or “free” Source format, and Ansi, or “fixed” Source format, see the table below:

Free Format Source	ANSI Format Source
There is no Sequence Number area.	The Sequence Number area is Columns 1-6. The compiler ignores the Sequence Number area.
The Indicator area is column 1	The Indicator area is column 7
Area A is columns 1-4	Area A is columns 8-11
Area B is columns 5 to end-of-line	Area B is columns 12-72
The Identication area starts after a « | » or « > » character that is not part of a literal expression.	The Identification area starts in column 73 and continues to end-of-line
Lines can be up to 1024 characters long	Lines can be up to 1024 characters long, but the compiler does not parse columns 1-6, or columns 73-end of line.

When a file is opened for editing, and the –free compiler flag is set, the Code Editor suppresses the Indicator Column in column 7, and for purposes of syntax coloring, and display in the Outline View, the source is parsed according to the Free Format source rules described above.

CVS/SubVersion support for remote projects

When working in a Windows environment, with a remote CVS repository, you can use the CVS Repository Explorer Perspective with the Developer Studio, to manage Source Code Control. This provides important functionality for the case where a team of Developers is using COBOL-IT Developer Studio in a local environment, managing source on a Server A using the Remote System Explorer, with a Source Code Control repository set up on a Server B.

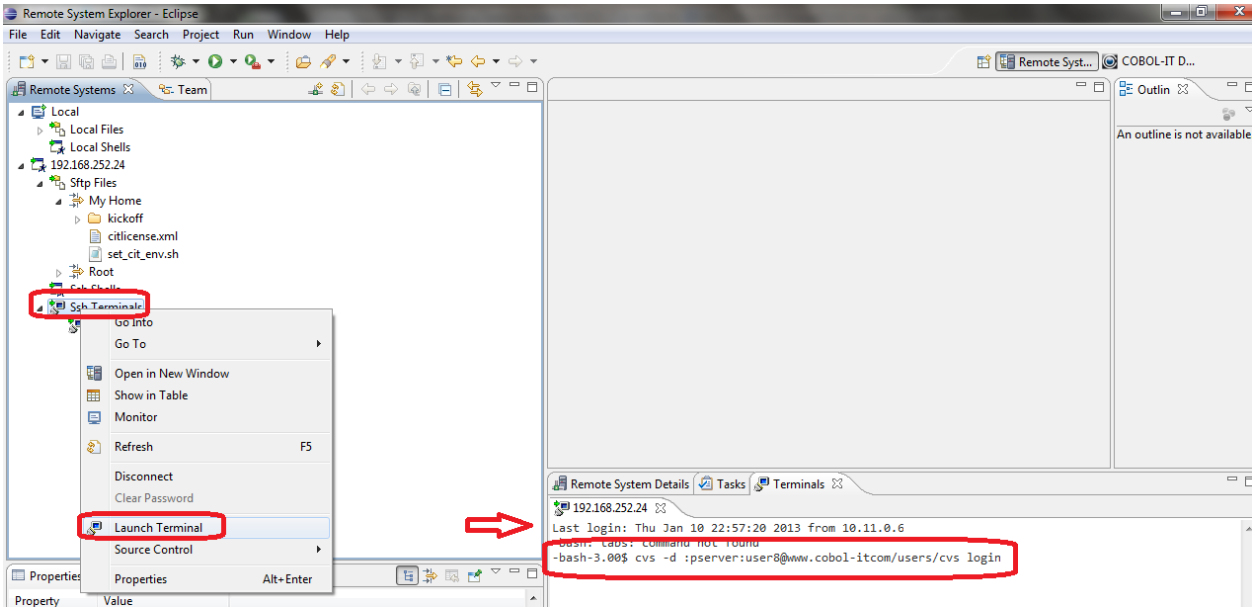


.
Prerequisites:
COBOL-IT Developer Studio 1.6.5 or later, Eclipse 3.7 or later

Guidelines for managing Source Code Control from within the Developer Studio

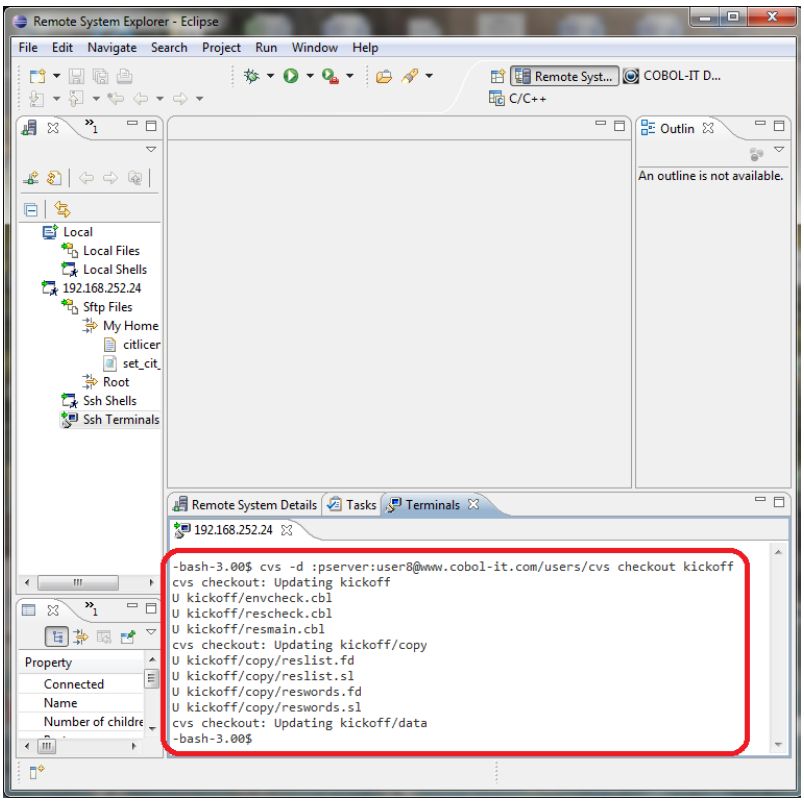
When using the Remote System Explorer, you must use the interfaces, and guidelines developed by COBOL-IT to manage Source Code Control.

Use the Terminal View in the Remote System Explorer Perspective to log into the Source Code Control Repository:



Example:
>cvs -d :pserver:user8@www.cobol-it.com/users/cvs login
[Password will be requested and stored by CVS]

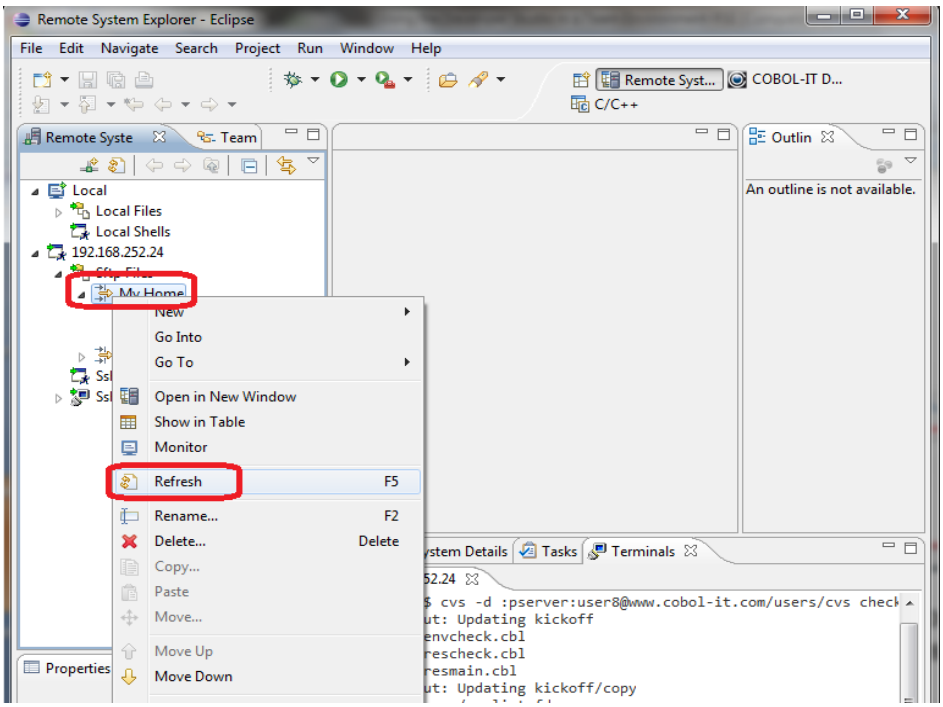
Use the Terminal View in the Remote System Explorer Perspective to checkout your project from the Source Code Control Repository:



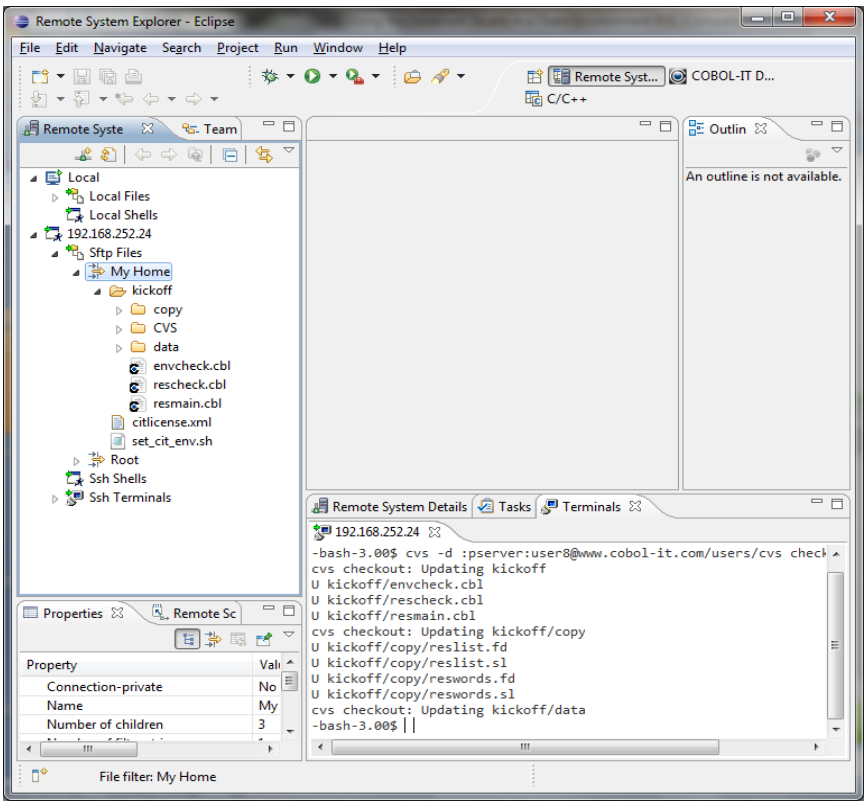
Example:
>cvs -d :pserver:user8@www.cobol-it.com/users/cvs checkout kickoff

This will create a folder under the Current Working Directory. Refresh the view on your home directory, and view the checked out project files:

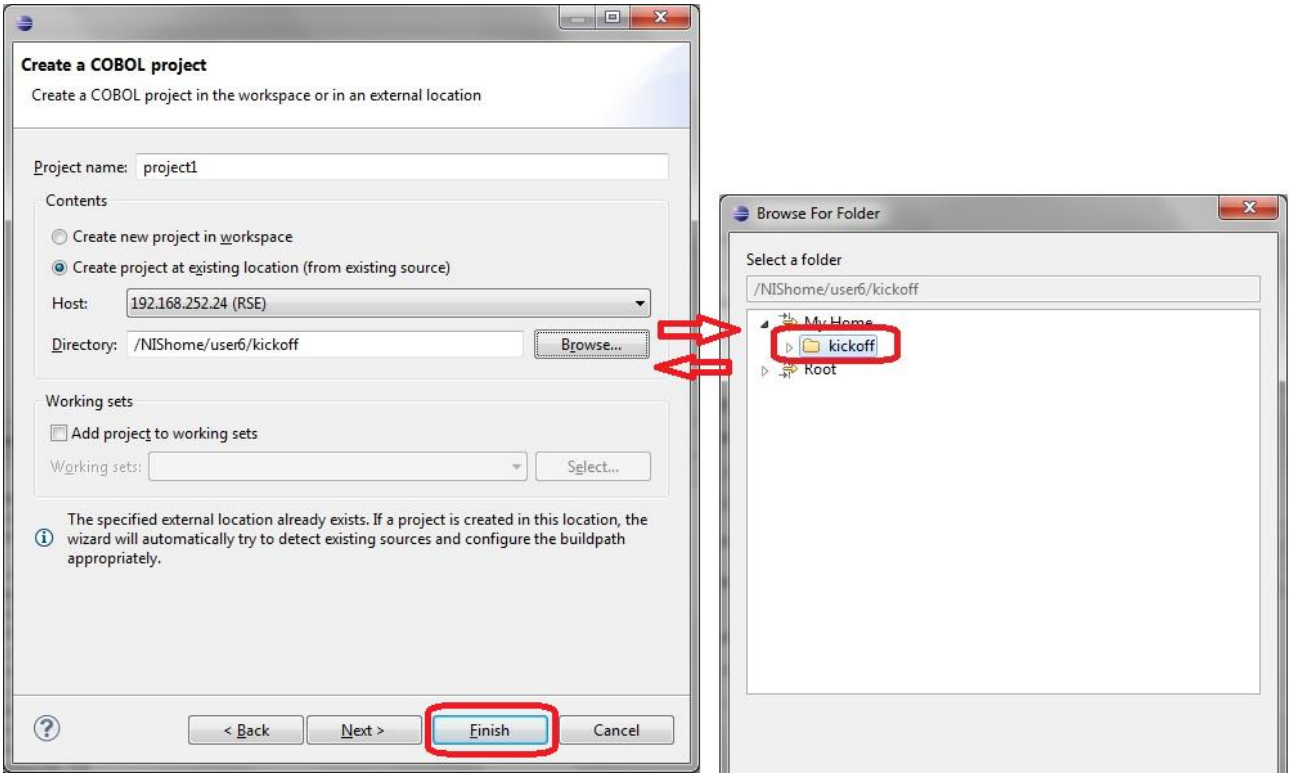
Example:



View the source that has been checked out:


Remote System Explorer - Eclipse
File Edit Navigate Search Project Run Window Help
Remote Syst...
COBOL-IT D...
C/C++
Remote Syste
Team
Local
Local Files
Local Shells
192.168.252.24
Sftp Files
My Home
kickoff
copy
CVS
data
envcheck.cbl
rescheck.cbl
resmain.cbl
citlicense.xml
set_cit_env.sh
Root
Ssh Shells
Ssh Terminals
Outlin
An outline is not available.
Remote System Details
Tasks
Terminals
192.168.252.24
-bash-3.00$ cvs -d :pserver:user8@www.cobol-it.com/users/cvs check
cvs checkout: Updating kickoff
U kickoff/envcheck.cbl
U kickoff/rescheck.cbl
U kickoff/resmain.cbl
cvs checkout: Updating kickoff/copy
U kickoff/copy/reslist.fd
U kickoff/copy/reslist.sl
U kickoff/copy/reswords.fd
U kickoff/copy/reswords.sl
cvs checkout: Updating kickoff/data
-bash-3.00$
Properties
Remote Sc
Property
Connection-private
Name
Number of children
3
File filter: My Home

Create a Project Using Existing Source



The project you have created contains all the elements necessary to manage your source using source code control.

The Source Control Interfaces

The Source Control tabs:
Window>Preferences>COBOL>Compiler>Source Control tab
Project>Properties>COBOL Properties>Source Control tab

The Source Control tab allows the user to enter source control commands for CVS, SVN, or Other, for a range of source control commands. Default settings are provided for CVS and SVN.

Command Defaults set for CVS

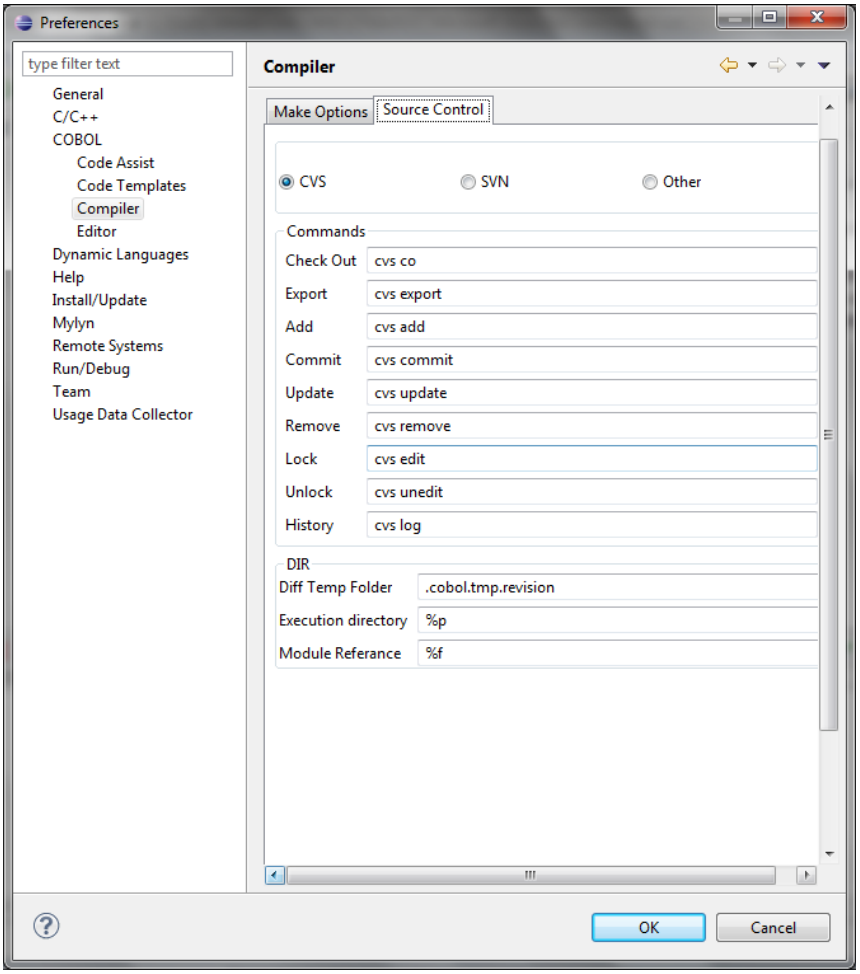
Function	Default setting CVS	Default setting SVN	Other setting	Remarks
Checkout	cvs co	Svn co	Not supported	Check source out of the repository, with CVS directories and files
Export	cvs export	Svn export	Not supported	Export source from the repository, stripped of CVS directories and files.
Add	cvs add	Svn add	Enter your	Add a source file

			command here	to the repository
Commit	cvs ci	Svn ci	Enter your command here	Commit, or Check source in to the repository.
Update	cvs update	Svn update	Enter your command here	Retrieve an updated source file from the repository
Remove	cvs remove	Svn remove	Enter your command here	Remove a source file from the repository. Must be followed by the Commit.
Lock	cvs edit	Svn lock	Enter your command here	Lock a file for exclusive use
Unlock	cvs unedit	Svn unlock	Enter your command here	Unlock a file that has been locked for exclusive use
History	cvs log	Svn log	Not supported	Retrieve the History Log for a file from the repository.

Command Defaults set for CVS

Diff Temp Folder	.cobol.tmp.revision	.cobol.tmp.revision	
Execution Directory	%p	%p	Module absolute path
Module Reference	%f	%f	Module file name (no path)

The Compiler > Source Control Interface

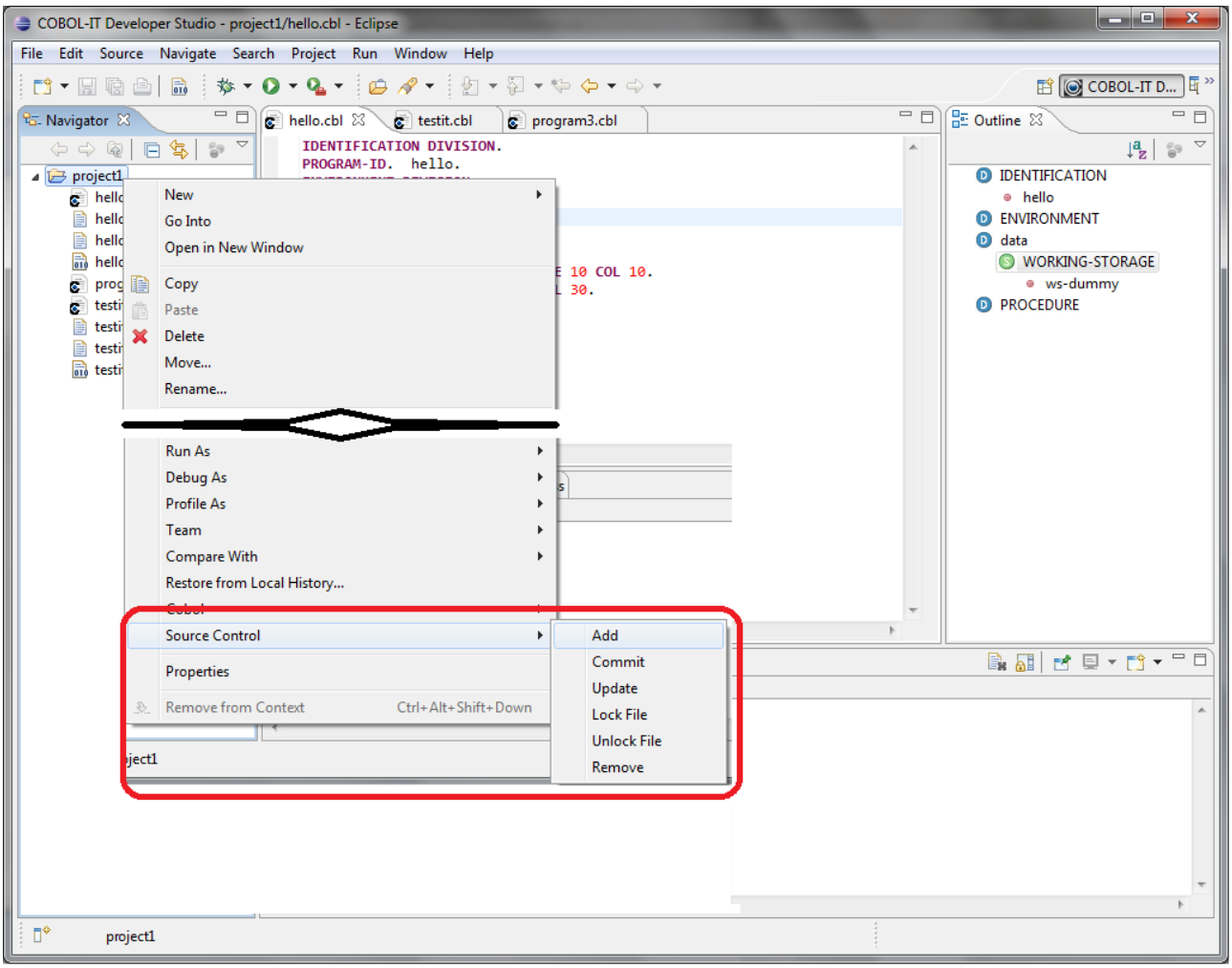


The Source Control tabs

Right-click Dropdown Menu on Project>Source Control

The Right-click Dropdown Menu on the Project allows the user to select one of the supported Source Control functions at the Project level:

Function	Default setting CVS	Default setting SVN	Other setting	Remarks
Add	cvs add	Svn add	Enter your command here	Add a source file to the repository
Commit	cvs ci	Svn ci	Enter your command here	Commit, or Check source in to the repository.
Update	cvs update	Svn update	Enter your command here	Retrieve an up-dated source file from the reposi-tory
Lock	cvs edit	Svn lock	Enter your command here	Lock a file for exclusive use
Unlock	cvs unedit	Svn unlock	Enter your command here	Unlock a file that has been locked for exclusive use
Remove	cvs remove	Svn remove	Enter your command here	Remove a source file from the re-pository. Must be followed by the Commit.

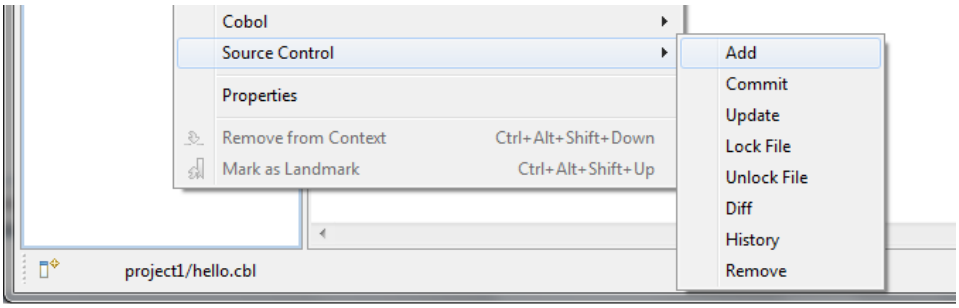


The Source Control tabs:
Right-click Dropdown Menu on [Source File]>Source Control

The Right-click Dropdown Menu on the Source File allows the user to select one of the supported Source Control functions at the File level:

Function	Default setting CVS	Default setting SVN	Other setting	Remarks
Add	cvs add	Svn add	Enter your command here	Add a source file to the repository
Commit	cvs ci	Svn ci	Enter your command here	Commit, or Check source in to the repository.
Update	cvs update	Svn update	Enter your command here	Retrieve an up-dated source file from the reposi-tory
Lock File	cvs edit	Svn lock	Enter your command here	Lock a file for exclusive use
Unlock File	cvs unedit	Svn unlock	Enter your command here	Unlock a file that has been locked for exclusive use
Diff	REPFILE='cat CVS/Reposito-ry'/hello.cbl; cvs			Compares a ver-sion of the file

	co -d tmpfolder -r 1.6 $REPFILE; U tmpfolder/hello.cbl 'tmpfolder/hello.cbl' -> '//usr/bob/project1/.cobol.tmp.revision/hello.cbl			from the repository with your working file
History	cvs log	Svn log	Not supported	Retrieve the History Log for a file from the repository.
Remove	cvs remove	Svn remove	Enter your command here	Remove a source file from the repository. Must be followed by the Commit.



Fixes

Window "Expression" causes delay while debugging

ID: 1145859668

When the Expression View was open in the Debugger Perspective, the performance of the debugger was poor.
This has been corrected.

Debugging with REPLACE statements and COPY statements

ID: 1145897224

When a COPY REPLACING statement modified source code statements in the Procedure Division at the time of compilation, the debugger did not correctly represent the source.

This has been corrected. The debugger now displays the code as compiled after the REPLACING operation. The results of the code translation after the application of the REPLACING operation are stored in subfolder c in [filename].i.

END PROGRAM in source causes IDE/Deet debugger to fail
ID: 1145897224

A source file ending with the line END PROGRAM. Caused both Deet, and the Developer Studio Debugger operate as though the program were not compiled for debugging.

This has been corrected.