

**COBOL-IT® Developer Studio**  
**Getting Started**  
**The Utilities**  
**Version 2.0**



## Contents

<b>ACKNOWLEDGMENT .....</b>	<b>6</b>
<b>COBOL-IT DEVELOPER STUDIO TOPICS.....</b>	<b>7</b>
<b>Introduction and License terms.....</b>	<b>7</b>
COBOL-IT Developer Studio License terms.....	7
<b>Dependencies .....</b>	<b>10</b>
Dependencies and Comments .....	10
The COBOL-IT Developer Studio Distribution.....	10
<b>THE DATA DISPLAYER .....</b>	<b>11</b>
<b>Data Displayer Configuration.....</b>	<b>12</b>
COBOL>DevOps Tools>Data Displayer .....	12
<b>Open the Data Displayer Perspective.....</b>	<b>13</b>
<b>The Data Displayer interface .....</b>	<b>13</b>
Select Host .....	14
File data source .....	14
Server path .....	14
Setup Script.....	15
Setup Script changes for EXTFH-compliant file systems.....	15
*.xdd file .....	16
Trace File .....	17
Organization code .....	18
Format.....	18
Encoding .....	18
Start Key .....	19
<b>The Data Displayer Toolbar .....</b>	<b>19</b>
Open mode.....	19
Select the View Mode.....	20
Select the Start Key.....	20
Select the START parameters.....	21
Select the Encoding & Click to view data .....	21
<b>The File Records Area .....</b>	<b>22</b>
Records Displayed in Tabular Form .....	22
Record Displayed in Single Record Mode.....	22
<b>The File Records Area Toolbar- Read Only Mode.....</b>	<b>23</b>
Read Previous/Read Next .....	23
Configure Columns .....	24
Close File .....	25
Re-open File.....	25
<b>Using Read/Write Mode.....</b>	<b>25</b>
The File Records Area Toolbar- Read-Write Mode.....	26
Add a Record .....	26
Delete a Record.....	26

Modify a Record (ASCII) .....	27
Modify a Data Element (Hex).....	27
Save Changes .....	28
Close File .....	29
<b>CODE COVERAGE.....</b>	<b>30</b>
<b>Quick Start Guide.....</b>	<b>30</b>
Enable COBOL code coverage in Project>Properties .....	30
Build the Project .....	31
Run the program .....	31
View results .....	32
<b>Overview.....</b>	<b>32</b>
<b>Launching in Coverage Mode.....</b>	<b>33</b>
Source Code Annotations.....	33
Decorators .....	34
<b>The Coverage View.....</b>	<b>36</b>
Opening the Coverage View .....	36
Coverage View Summaries.....	36
The Coverage View Toolbar.....	37
<b>Managing Coverage Sessions.....</b>	<b>38</b>
Session Lifecycle .....	38
Relaunch Coverage Session .....	38
Link with Current Selection.....	39
The Active Session .....	39
Select Active Session .....	39
Merging Sessions .....	40
<b>Session Import and Export .....</b>	<b>41</b>
Coverage Session Import .....	41
Coverage Session Export .....	43
<b>PROFILER.....</b>	<b>45</b>
<b>Profiling Preferences .....</b>	<b>45</b>
<b>Runtime Environment Variables .....</b>	<b>46</b>
COB_PROFILING_DIR.....	46
COB_PROFILING_EACH_MODULE.....	47
<b>The Profiler View.....</b>	<b>47</b>
Opening the Profiler View .....	47
Paragraphs Tab .....	48
Runtime Tab .....	49
The Profiler View Toolbar.....	49
Relaunch Profiler Session .....	50
Select Active Session .....	50
Link with Current Selection.....	50
<b>Profiler Session Import .....</b>	<b>50</b>
<b>Profiler Session Export.....</b>	<b>52</b>

<b>USING GIT FOR SOURCE CODE CONTROL.....</b>	<b>53</b>
<b>Configure Team Settings.....</b>	<b>54</b>
Window>Preferences>Team>Git .....	54
Window>Preferences>Team>Git>Configuration .....	54
Window>Preferences>Team>Models .....	55
<b>Open the Git Perspective.....</b>	<b>55</b>
<b>The Git Perspective.....</b>	<b>56</b>
<b>Create a New Repository.....</b>	<b>57</b>
Configure the new Git Repository .....	57
Your Project in the Developer Studio Perspective .....	58
Your Project in the Git Perspective.....	59
<b>Clone a Git Repository .....</b>	<b>60</b>
Source Git Repository.....	60
Branch Selection .....	61
Local Destination.....	62
The Cloned project in the Git Perspective .....	62
Git Repositories>Import Project .....	63
Access the Project in the Developer Studio Perspective.....	64
<b>Add an existing Git Repository.....</b>	<b>65</b>
Git Repositories View>Import Project.....	66
Access the Project in the Developer Studio Perspective.....	66
<b>Team&gt;Commit .....</b>	<b>67</b>
Commit Code to the Repository.....	67
Git Staging .....	67
<b>Git Repositories View .....</b>	<b>69</b>
Switch to>New Branch .....	69
Show in > History View .....	70
<b>USING RSE GIT FOR SOURCE CODE CONTROL .....</b>	<b>71</b>
<b>Setup for use of RSEGit .....</b>	<b>71</b>
Remote Server>COBOL Source Folder.....	71
Remote Server>COBOL Source Folder>git init.....	71
Developer Studio>Window>Preferences>Team>Models .....	71
Remote System Explorer>Establish a remote connection .....	72
Developer Studio>Create a project at existing location .....	73
Developer Studio>Open the RSEGit Perspective .....	76
RSEGit>RSEGit Repositories View .....	76
<b>Committing Files.....</b>	<b>77</b>
The Navigator View of the COBOL Project>Untracked text .....	77
[Project]>Team>Commit>RSEGit Staging .....	78
The Navigator View of the COBOL Project>Tracked text.....	82
<b>Create a New Branch.....</b>	<b>83</b>
Show In History .....	84
<b>Importing Source Files into the Project.....</b>	<b>84</b>





<b>Add an existing RSEGit Repository .....</b>	<b>87</b>
Show In History .....	88
 <b>USING MYLYN FOR TASK MANAGEMENT .....</b>	 <b>90</b>
<b>Working with a Remote Task Repository .....</b>	<b>90</b>
Select a task repository type.....	91
Github Issues.....	91
Enter query parameters .....	92
Task Repository View.....	92
 <b>Working with a Task List .....</b>	 <b>92</b>
Task List View .....	93
Handling an Existing Task.....	93
Reply to the Task .....	94
Submit Reply to the Task.....	94
 <b>Working with a Local Task Repository .....</b>	 <b>94</b>
Create a new local task .....	94
Open a Local Task .....	95
Close a Local Task.....	96

## Acknowledgment

**Copyright 2008-2020 COBOL-IT S.A.R.L. All rights reserved. Reproduction of this document in whole or in part, for any purpose, without COBOL-IT's express written consent is forbidden.**

Microsoft and Windows are registered trademarks of the Microsoft Corporation. UNIX is a registered trademark of the Open Group in the United States and other countries. Other brand and product names are trademarks or registered trademarks of the holders of those trademarks.

### **Contact Information:**

COBOL-IT  
231, rue Saint-Honoré - 75001 Paris - FRANCE  
Tel.: +33 1 75 43 05 50  
Fax: +33 1 75 43 05 16  
Email: [contact@cobol-it.com](mailto:contact@cobol-it.com)  
[www.cobol-it.com](http://www.cobol-it.com)

# COBOL-IT Developer Studio Topics

## Introduction and License terms

This document describes how to use the **COBOL-IT Developer Studio Data Displayer**. **COBOL-IT Developer Studio** is COBOL-IT's eclipse-based development environment, designed to support users of the **COBOL-IT Compiler Suite**.

## COBOL-IT Developer Studio License terms

The copyright for the COBOL-IT Developer Studio® is wholly owned by COBOL-IT. Any unauthorized reproduction of the software without the express written consent of COBOL-IT is prohibited.

For more information, please contact us at: [contact@cobol-it.com](mailto:contact@cobol-it.com)

COBOL-IT Corporate Headquarters are located at

231, rue Saint-Honore  
75001 Paris  
Tel: +33.1.75.43.05.15  
Email: [contact@cobol-it.com](mailto:contact@cobol-it.com)

COBOL-IT, COBOL-IT Compiler Suite, CitSQL, CitSORT, and COBOL-IT Developer Studio are trademarks or registered trademarks of COBOL-IT.

Eclipse is a trademark of the Eclipse Foundation.

IBM, and AIX are registered trademarks of International Business Machines Corporation.

Linux is a registered trademark of Linus Torvalds.

Windows, Visual Studio, and Visual Studio Express are registered trademarks of Microsoft Corporation.

Java and Solaris are registered trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark of The Open Group

HP is a registered trademark of Hewlett Packard, Inc.

Red Hat is a registered trademark of Red Hat, Inc.

All other trademarks are the property of their respective owners.



<b>COBOL-IT DEVELOPER STUDIO TOPICS.....</b>	<b>2</b>
Introduction and License terms.....	7
Dependencies.....	10
<b>THE DATA DISPLAYER .....</b>	<b>11</b>
Data Displayer Configuration.....	12
Open the Data Displayer Perspective.....	13
The Data Displayer interface .....	13
The Data Displayer Toolbar .....	19
The File Records Area.....	22
The File Records Area Toolbar- Read Only Mode.....	23
Using Read/Write Mode.....	25
<b>CODE COVERAGE.....</b>	<b>30</b>
Quick Start Guide.....	30
Overview.....	32
Launching in Coverage Mode.....	33
The Coverage View.....	36
Managing Coverage Sessions.....	38
Session Import and Export .....	41
<b>PROFILER.....</b>	<b>45</b>
Profiling Preferences .....	45
Runtime Environment Variables .....	46
The Profiler View.....	47
Profiler Session Import .....	50
Profiler Session Export.....	52
<b>USING GIT FOR SOURCE CODE CONTROL.....</b>	<b>53</b>
Configure Team Settings.....	54
Open the Git Perspective.....	55





---

The Git Perspective.....	56
Create a New Repository.....	57
Clone a Git Repository .....	60
Add an existing Git Repository.....	65
Team>Commit .....	67
Git Repositories View .....	69
<b>USING RSE GIT FOR SOURCE CODE CONTROL .....</b>	<b>71</b>
Setup for use of RSEGit .....	71
Committing Files.....	77
Create a New Branch.....	83
Importing Source Files into the Project.....	84
Add an existing RSEGit Repository .....	87
<b>USING MYLYN FOR TASK MANAGEMENT.....</b>	<b>90</b>
Working with a Remote Task Repository .....	90
Working with a Task List .....	92
Working with a Local Task Repository .....	94



## Dependencies

### Dependencies and Comments

Dependency	Comment
“C” compiler	The COBOL-IT Compiler requires a “C” compiler. While most Linux>Unix installations will include a “C” compiler, many Windows installations will not. Windows users can download the Visual Studio from <a href="http://www.microsoft.com">www.microsoft.com</a> .
COBOL-IT Compiler Suite	The COBOL-IT Compiler Suite, Standard Edition can be downloaded at the COBOL-IT Online Portal. For access to the COBOL-IT Online Portal, please contact your sales representative at <a href="mailto:sales@cobol-it.com">sales@cobol-it.com</a> .
Java Runtime Environment (JRE)	The COBOL-IT Developer Studio Kepler build can be run with the Java Runtime Environment (JRE) Version 1.6 or greater. The COBOL-IT Developer Studio Neon build can be run with the JRE Version 1.8 or greater.
Eclipse	Eclipse is included with the download of Developer Studio.

The COBOL-IT Developer Studio requires that the COBOL-IT Compiler Suite already be installed on the host platform, and that a “C” compiler be installed on the host platform.

The COBOL-IT Developer Studio is an Eclipse plug-in, and as such, requires that Eclipse be installed on the host platform. Eclipse, in turn, requires that a Java Runtime Environment (JRE) be installed on the host platform.

### The COBOL-IT Developer Studio Distribution

For Windows-based installations, the COBOL-IT Developer Studio, Enterprise Edition can be downloaded from the COBOL-IT online portal with a login and password provided by your sales representative.

The COBOL-IT Developer Studio, Enterprise Edition is available with Subscription. The COBOL-IT Developer Studio, Enterprise Edition provides functionality with the installation of several Perspectives:

- Developer Studio Perspective in which users set up and build COBOL projects, using a locally installed version of the COBOL-IT Compiler Suite Enterprise Edition. The Developer Studio Perspective additionally provides access to Code Coverage and Profiling Tools.
- Debugger Perspective providing access to a feature-rich COBOL debugger both locally, and on Remote Systems
- Remote Systems Perspective, allowing use of Compiler, Runtime, and Debugger functionalities installed on remote servers.





- Git and RSEGit Perspectives, providing users with full access to the Git/Github Source Code Control System.
- Data Displayer Perspective, providing access to a tool for browsing and modifying data in indexed, sequential and relative files.
- Planning Perspective, providing access to the Mylyn Task Manager.
- For more information about the usage of Git/RSEGit, Data Displayer, Mylyn Task Manager, and Code Coverage, see the Getting Started with the Developer Studio- The Utilities Manual.
- Using the COBOL-IT Developer Studio requires a license for both the COBOL-IT Compiler Suite Enterprise Edition, and COBOL-IT Developer Suite.

## The Data Displayer

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. The Data Displayer can be used to browse and edit indexed files. Line sequential, binary sequential and relative files can be browsed. 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 (**XDDs**) 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.

The screenshot shows the COBOL-IT Data Displayer application window. The title bar reads "workspace - Cobol-IT Data Displayer - project1/holidaysIX.cbl - COBOL-IT Developer Studio". The menu bar includes File, Edit, Source, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations. The main window is divided into two panes. The left pane, titled "Data Displayer", contains a "Settings" section with fields for "Select host" (Localhost), "File data source" (C:\COBOL\COBOLIT\DevStudio200\workspace\project1\data\holidaysIX.dat), "Server path" (C:\COBOL\COBOLIT\bin\dd\_extfh.exe), "Setup script" (C:\COBOL\COBOLIT\setenv\_cobolit.bat), "\*xdd file" (C:\COBOL\COBOLIT\DevStudio200\workspace\project1\HOLIDAYSIX.xdd), and "Trace file" (coberr.txt). The right pane, titled "File Records: HOLIDAYSIX", displays a table of holiday records. The table has columns for HOLIDAY\_NAME, HOLIDAY\_DATE\_WE..., HOLIDAY\_DATE\_TH..., HOLIDAY\_DATE\_TH..., HOLIDAY\_DATE\_TH..., HOLIDAY\_CURRENT..., HOLIDAY\_CURRENT..., and HOLIDAY\_CURRENT... The table contains 24 rows of data, including entries for ALL Saints Day, Christmas, Columbus Day, Day-After Christ..., Day-After Thanks..., Fathers Day, Halloween, Independence Day..., Labor Day, Martin Luther Ki..., Memorial Day, Mothers Day, New Year's Eve, New Years Day, Presidents Day, St Patrick's Day..., Thanksgiving, and Valentines Day.

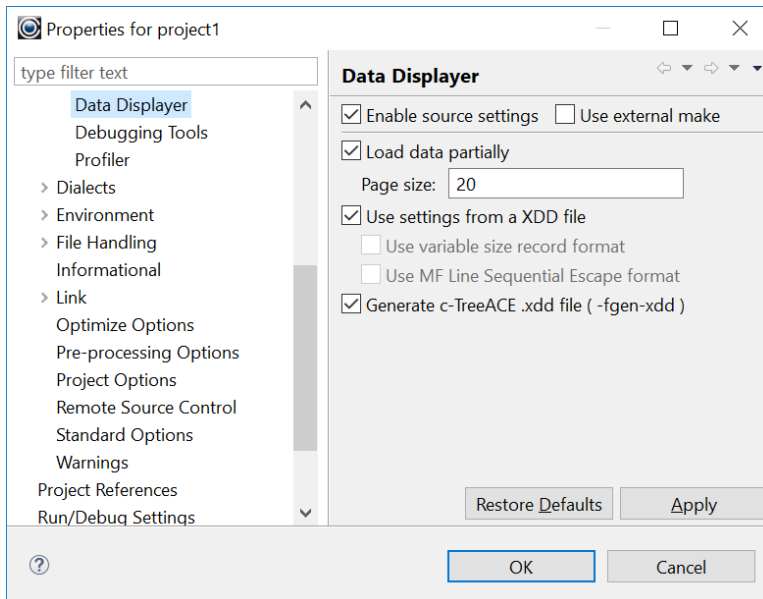
HOLIDAY_NAME	HOLIDAY_DATE_WE...	HOLIDAY_DATE_TH...	HOLIDAY_DATE_TH...	HOLIDAY_DATE_TH...	HOLIDAY_CURRENT...	HOLIDAY_CURRENT...	HOLIDAY_CURRENT...
ALL Saints Day	Wednesday	November	001	2017	20171228	14183306	-0500
Christmas	Monday	December	025	2017	20171228	14183306	-0500
Columbus Day	Monday	October	009	2017	20171228	14183306	-0500
Day-After Christ...	Tuesday	December	026	2017	20171228	14183306	-0500
Day-After Thanks...	Friday	November	024	2017	20171228	14183306	-0500
Fathers Day	Sunday	June	018	2017	20171228	14183306	-0500
Halloween	Tuesday	October	031	2017	20171228	14183306	-0500
Independence Day...	Tuesday	July	004	2017	20171228	14183306	-0500
Labor Day	Monday	September	004	2017	20171228	14183306	-0500
Martin Luther Ki...	Monday	January	016	2017	20171228	14183306	-0500
Memorial Day	Monday	May	029	2017	20171228	14183306	-0500
Mothers Day	Sunday	May	014	2017	20171228	14183306	-0500
New Year's Eve	Sunday	December	031	2017	20171228	14183306	-0500
New Years Day	Sunday	January	001	2017	20171228	14183306	-0500
Presidents Day	Monday	February	020	2017	20171228	14183306	-0500
St Patrick's Day...	Friday	March	017	2017	20171228	14183306	-0500
Thanksgiving	Thursday	November	023	2017	20171228	14183306	-0500
Valentines Day	Tuesday	February	014	2017	20171228	14183306	-0500





## Data Displayer Configuration

### COBOL>DevOps Tools>Data Displayer



Here, the behavior of the Table Display configured, by setting the size of the table page. In our example, we have set the table page size to 100. As a result, when we do a tabular display, 100 records will be read into the table, and a vertical scroll bar will allow us to browse these records. When we reach the end of these 100 records, another 100 records will be read.

Typically, you will keep the defaults of reading settings from the XDD file, and compiling with -fgen-xdd.

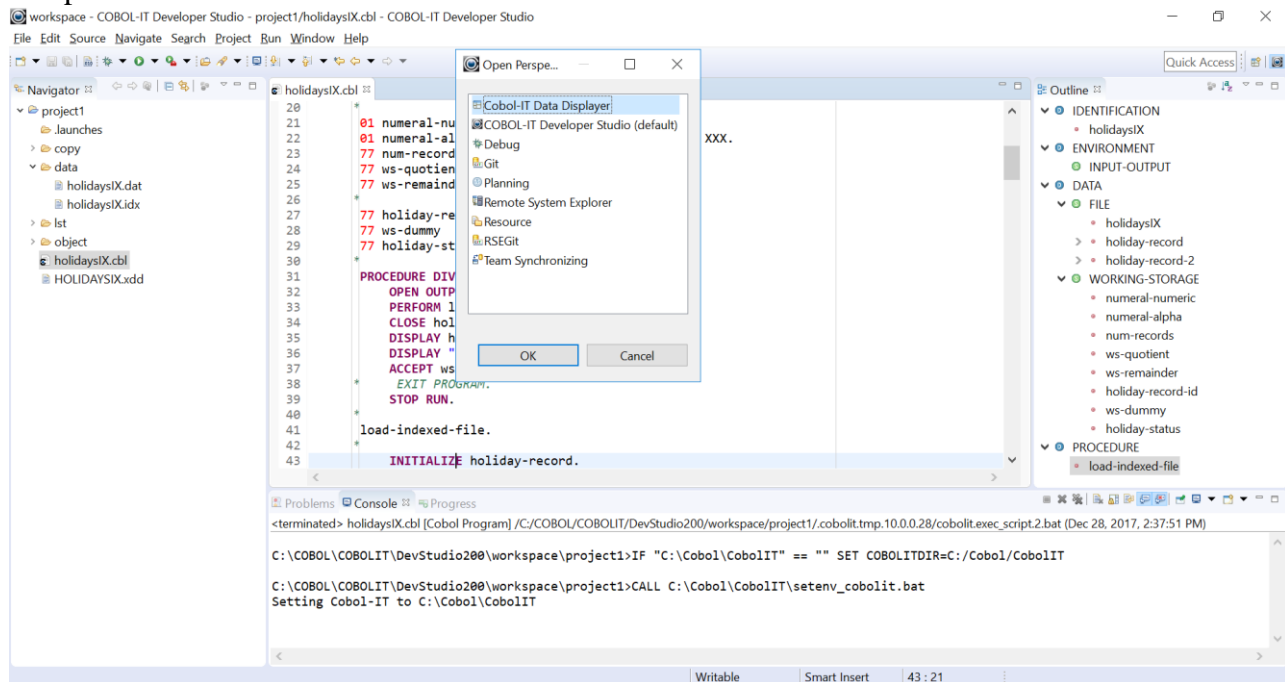
the Variable and Line Sequential Formats are critical when handling different types of data files. The Data Displayer recognizes variable size record formats.

When using Micro Focus Line Sequential files, the Data Displayer can be set to match the MF Line Sequential Escape format.



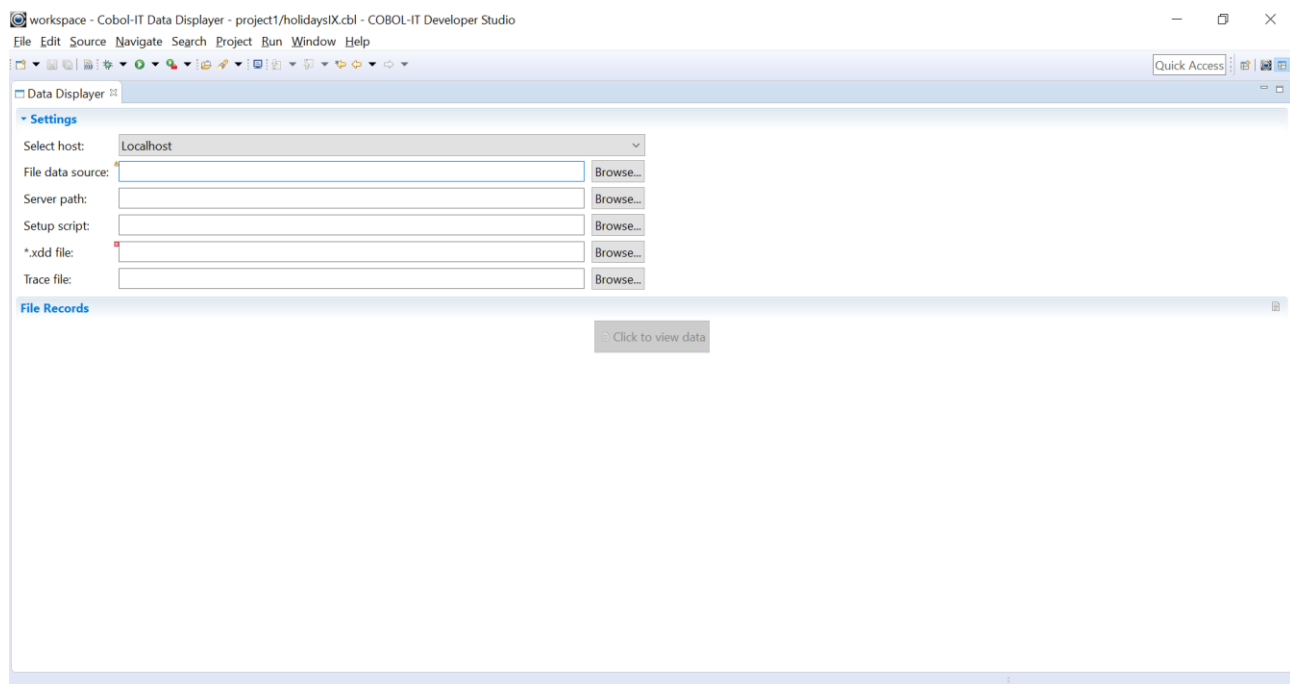
## Open the Data Displayer Perspective

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.



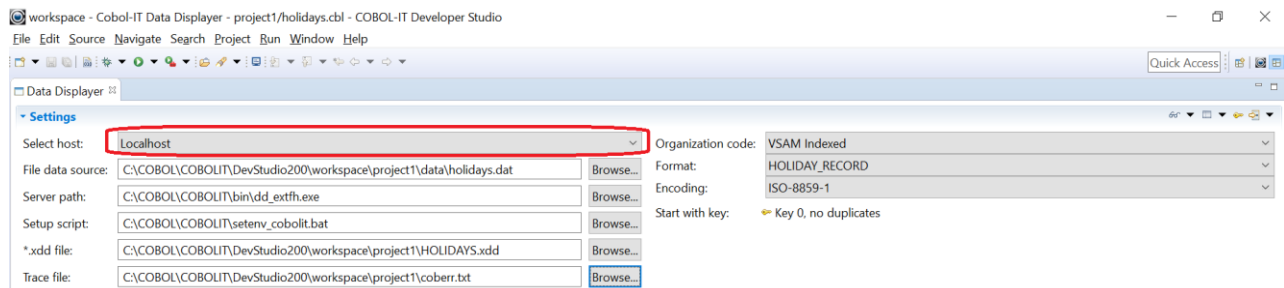
## The Data Displayer interface

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



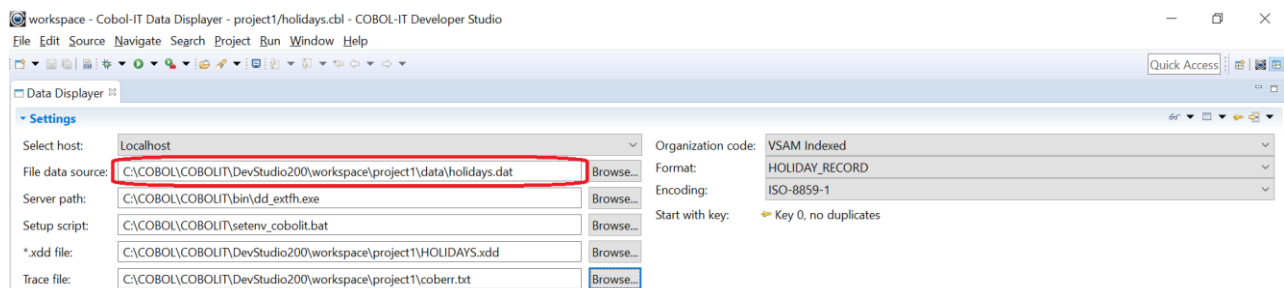
## Select Host

Dropdown the Select Host combo box, to see available local and remote connections. In our example, we will select Localhost. If a Remote System Explorer connection has been made, it will also be available in the dropdown box. In our examples, all source files, .xdd files, and data files are located on the Localhost.



## File data source

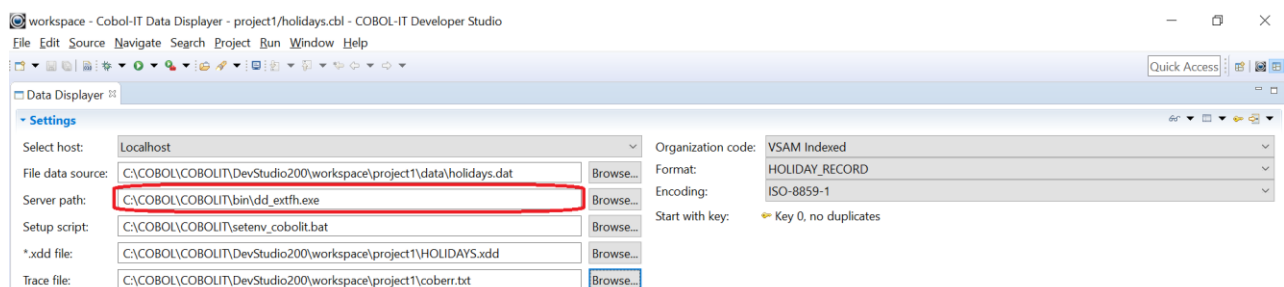
Click on the Browse button, and locate the data file you wish to display in the Data Displayer. From the Browse for File screen, select the file, and click OK to return to the Data Displayer.



## 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.

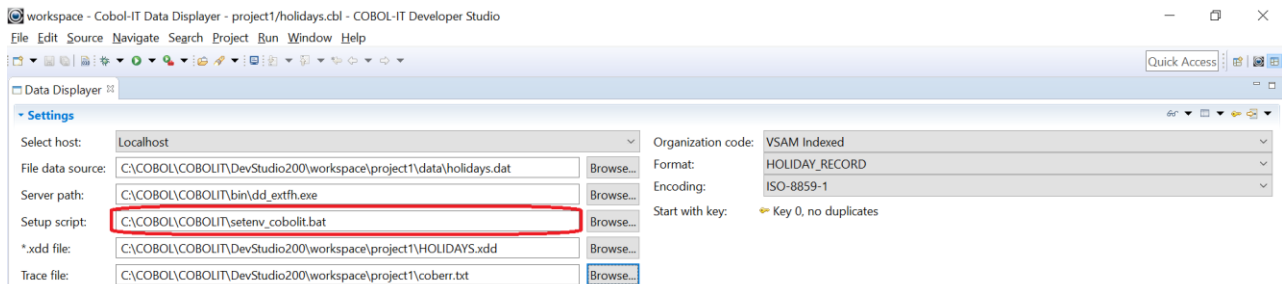
From the Browse for File screen, select the EXTFH server and click OK to return to the Data Displayer.





## Setup Script

On Windows platforms, the Setup Script is called `setenv_cobolit.bat` and is located in the installation directory. On Linux/UNIX platforms, the Setup script is called `cobol-it-setup.sh` and is located in the `bin` directory located under the installation directory.



## Setup Script changes for EXTFH-compliant file systems

When using the Data Displayer to view data in EXTFH-compliant file systems, the Data Displayer requires the same runtime environment as the COBOL-IT Enterprise Runtime. The most common cases will be the EXTFH-compliant file systems whose support has been facilitated by COBOL-IT, with the implementation of specific compiler flags and runtime environment variables.

### COBOL-IT includes EXTFH Libraries

The COBOL-IT distribution includes EXTFH drivers for the BerkeleyDB, D-ISAM and VBISAM file systems. The EXTFH drivers and libraries for the C-Tree ISAM File engine can be acquired from Faircom and C-Tree is also supported. Enabling the use of any of these file systems can be done either with the use of a compiler flag, or with a setting of the `COB_EXTFH` runtime environment variables.

File System	Compiler Flag	Compiler Config File	COB_EXTFH
BerkeleyDB	-fbdb	bdb:yes	COB_EXTFH=bdbextfh
D-ISAM	-fdisam	disam:yes	COB_EXTFH=disamextfh
C-Tree	-fctree	ctree:yes	COB_EXTFH=CTEXTFH COB_EXTFH_LIB=~/.ctree.cobol/extfh/CTEXTFH.so: /opt/cobol-it-64/lib/libcitextfh.so (Linux/UNIX) COB_EXTFH_LIB=~/.ctree.cobol/extfh/CTEXTFH.dll; C:\COBOL\COBOLIT\bin\citextfh_dll.dll (Windows)
VBISAM	-fvbisam	vbisam:yes	>export COB_EXTFH=vbisamextfh

When the `COB_EXTFH` environment variable is defined, the runtime looks for `lib$(COB_EXTFH)` on UNIX and `%COB_EXTFH%.dll.dll` on Windows in the COBOL-IT installation directory and all directories indicated in the `COB_LIBRARY_PATH`.

When using the Data Displayer with EXTFH-compliant file systems, the user has a few options.

When working on the LocalHost, you can launch the Developer Studio from a script in which you set the `COB_EXTFH` environment variable prior to launching the Developer Studio. As examples:





```
DevStudioDISAM.bat (Windows)
SET PATH=[path to CDS];%PATH%
SET COB_EXTFH=disamextfh
CDS.EXE
```

```
devstudio.sh (Linux/UNIX)
export PATH=[path to CDS]:$PATH
export COB_EXTFH=disamextfh
CDS
```

Alternatively, you can manage the settings of COB\_EXTFH through your COBOL-IT Setup Script.

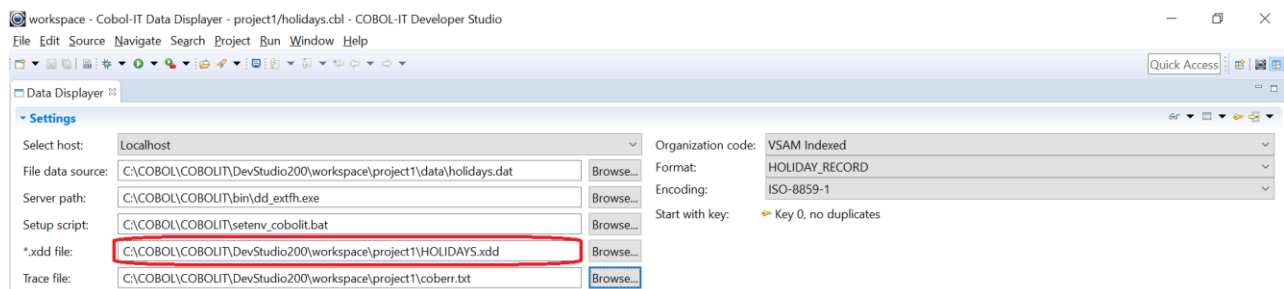
## \*.xdd file

### 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 a primary key and two alternate keys, and 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 the file for data display that the field names 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, and also how you use the selection of the key and the start function to alter the display of the data.





```
<?xml version="1.0" encoding="US-ASCII"?>
<table name="HOLIDAYSIX" maxRecLen="71" minRecLen="70" >
<key duplicate="false" primary="true" >
<part name="HOLIDAY_NAME" offset="0" size="25" />
</key>
<key duplicate="true" primary="false" >
<part name="HOLIDAY_DATE" offset="25" size="25" >
<part name="WEEK_DAY" offset="25" size="9" />
<part name="THE_MONTH" offset="34" size="9" />
<part name="THE_DAY" offset="43" size="3" />
<part name="THE_YEAR" offset="46" size="4" />
</part>
</key>
<select organization="2" lsmf="1" varrec="0" >
indexed
</select>
<schema name="HOLIDAY_RECORD" size="71" >
<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="3" type="NumUnsigned" digits="3" scale="0" />
<field name="HOLIDAY_DATE_THE_YEAR" offset="46" size="4" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD" offset="50" size="8" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS" offset="58" size="8" type="Alphanum" />
<field name="HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET" offset="66" 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.

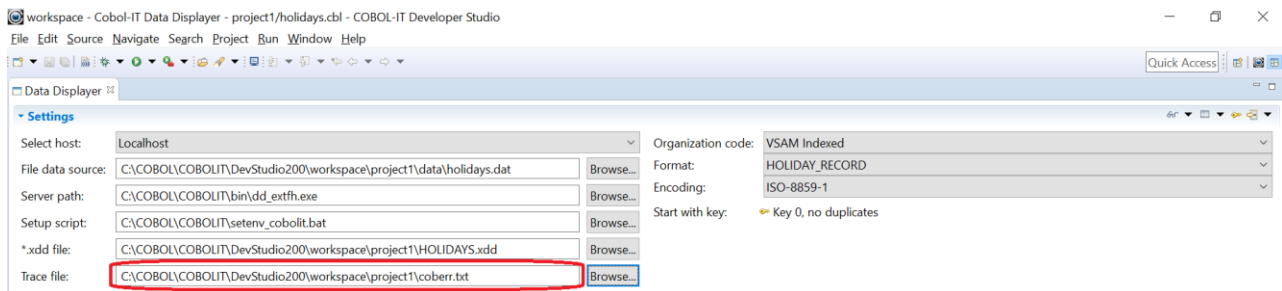
The Organization Code is set to VSAM Indexed. This information has been extracted from the XDD file.

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. The user can toggle between the views of these two formats by toggling between these formats on the Format combo-box. The Start key has been set to the primary key (Key 0, no duplicates).

## **Trace File**

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.

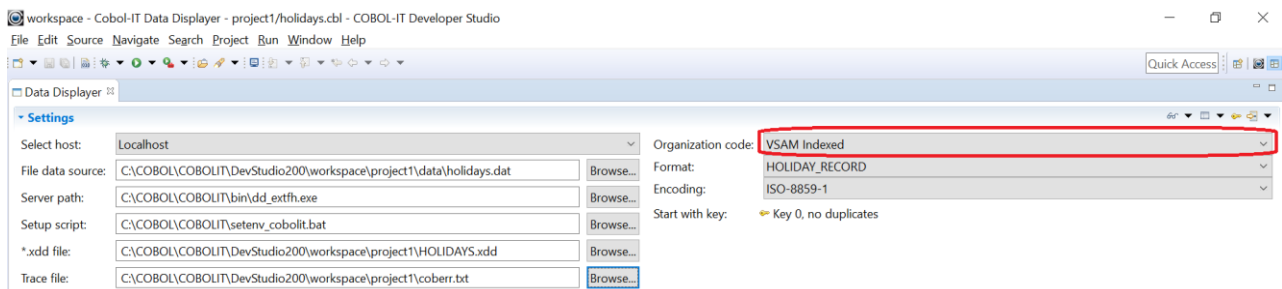




The trace file should be a line sequential file. If it does not exist, it will be created. Trace file information is renewed every time an Open function is performed.

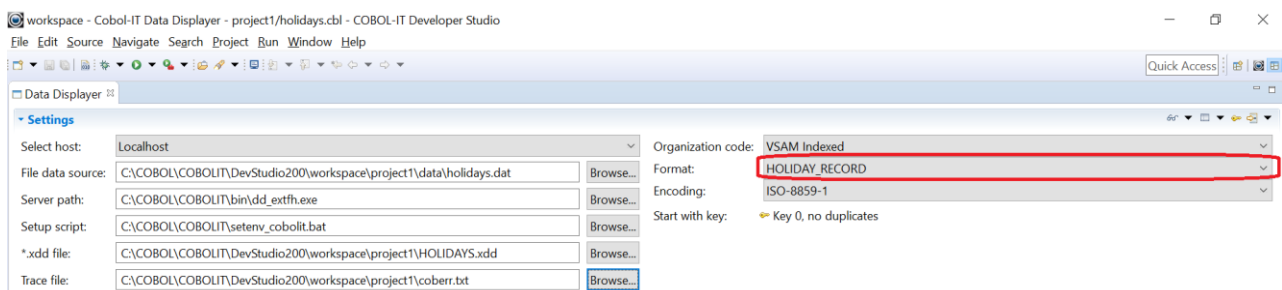
## Organization code

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

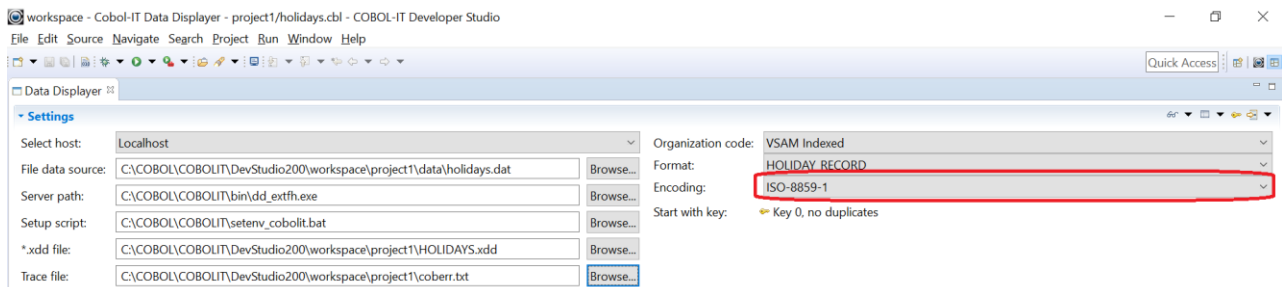
Our XDD has two record formats, and this is reflected in the choices available in the Format dropdown box. When a file is open in the Data Displayer, you can toggle between available data formats without closing and re-opening the file.



## Encoding

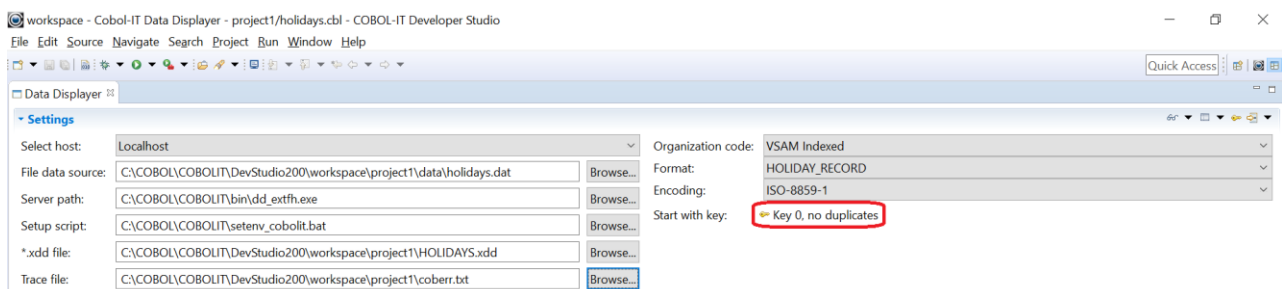
The Data Displayer derives the data encoding from the xdd file.





## Start Key

The Start key has been set to the primary key (Key 0, no duplicates).



## The Data Displayer Toolbar

### 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.

When you open a file in Read Only mode:

- You may view your file in Table Mode or in Single Record Mode.
- You may elect to read through your file on any key.
- If you are in Single-record mode, you may use the START functions.
- You may browse your data. You may not modify your data.

When you open a file in Read-Write mode:

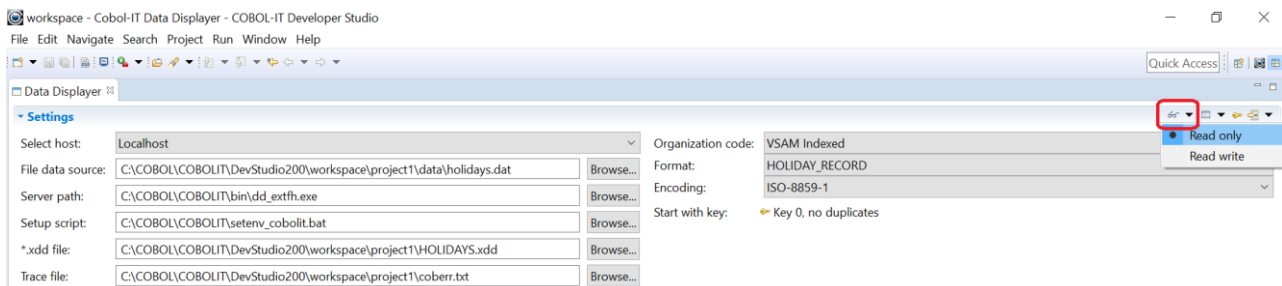
- You may only view your file in Single Record Mode
- You may elect to read through your file on any key
- You may use the START functions, and READ NEXT/READ PREVIOUS functions to isolate the record you wish to modify.
- You may not modify the primary key. If you wish to modify the primary key, you must delete the existing record, and add a new record with a new primary key.







Select the Mode “Read only” to OPEN the File.

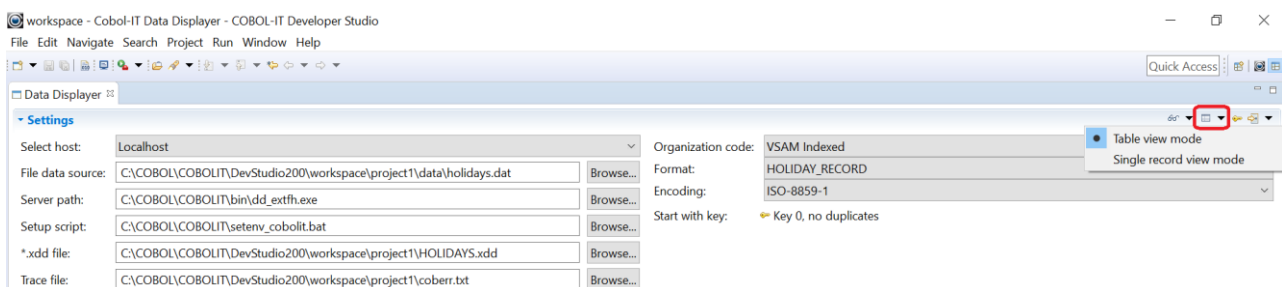


## Select the View Mode

Select whether you wish to view the records in “Table mode” or in “Single record mode”.

When you select Table view mode

- You may use the scroll-bar to modify the view of the file.
- You may not operate in READ-WRITE mode.
- If you change View Mode, you must re-load the file. If you also change the READ/READ-WRITE mode, you must CLOSE the file before you re-load it.
- If you change from READ to READ-WRITE mode, the Data Displayer will automatically set your file view in Single record mode. By default this would position your view of the file on the first record of the file. If you wish to modify a specific record, you must use the START function to position your view of the file on that record.
- If you change from READ-WRITE to READ mode, the Data Displayer will be in Single record mode by default. If you also wish to return to Table view mode, you must change this setting as well.

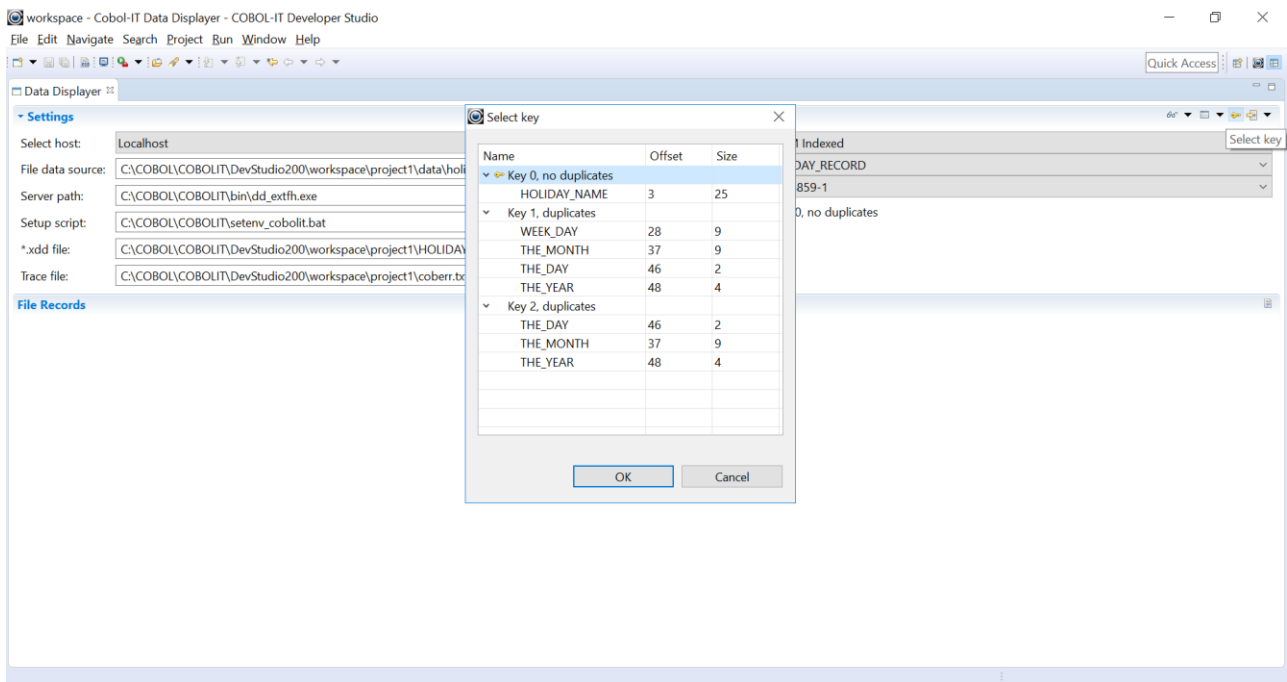


## Select the Start Key

Open the “Select Key” dialog window by clicking on the “Select key” icon. A key icon is displayed next to the current selected key. In this case, Key 0, the default key, is selected. Click OK.

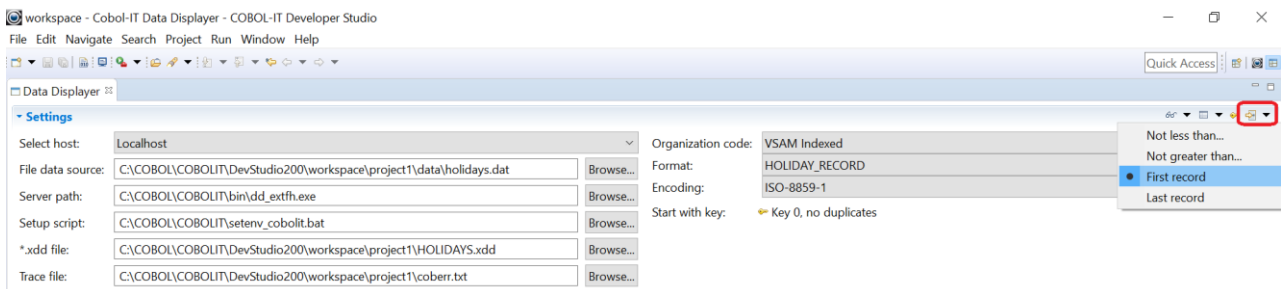






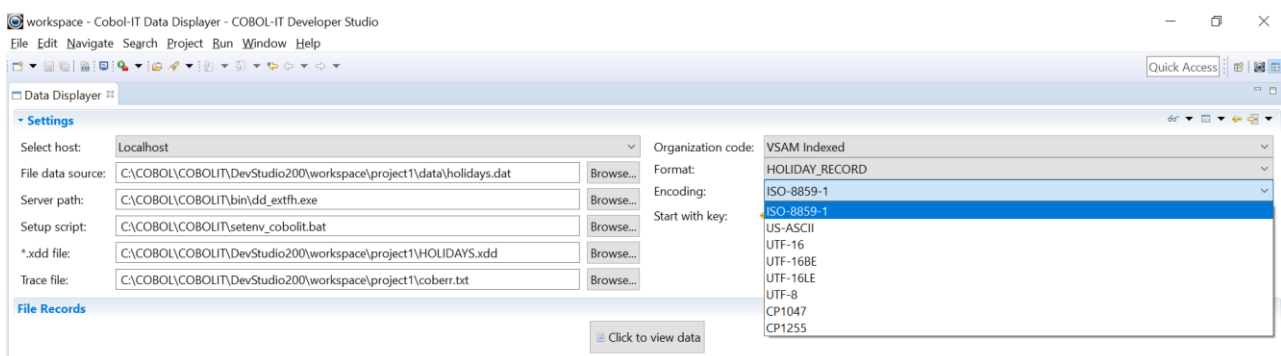
## Select the START parameters

Selecting a START parameter other than the default “First record” will place your View in Single record mode. For this exercise, where we have selected “Table view mode”, keep the default.



## Select the Encoding & Click to view data

ISO-8859-1 is the default. If you require an alternative, drop down the Encoding combo-box and select from othe available options.





Then select the “Click to view data” button to display the file in tabular format:  
The data displays, sorted on the primary key.

## The File Records Area

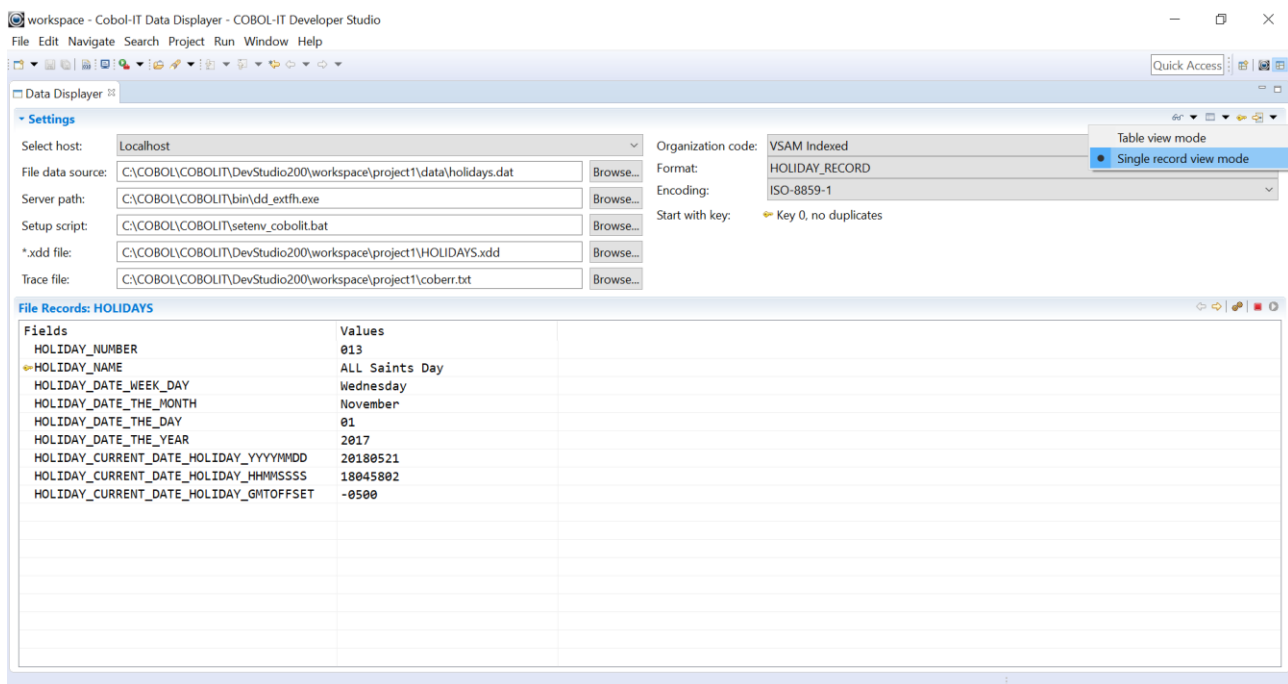
### Records Displayed in Tabular Form

The screenshot shows the COBOL-IT Developer Studio interface. The 'Data Displayer' window is open, displaying a tabular view of holiday records. The 'Settings' pane on the left shows the file path 'C:\COBOL\COBOLIT\DevStudio200\workspace\project1\data\holidays.dat' and other configuration options. The main area shows a table with columns for holiday names, dates, and times.

HOLIDAY_NUMBER	HOLIDAY_NAME	HOLIDAY_DATE_WE...	HOLIDAY_DATE_TH...	HOLIDAY_DATE_TH...	HOLIDAY_DATE_TH...	HOLIDAY_CURRENT...	HOLIDAY_CURRENT...	HOLIDAY...
013	ALL Saints Day ...	Wednesday	November	01	2017	20180521	18045802	-0500
017	Christmas ...	Monday	December	25	2017	20180521	18045802	-0500
011	Columbus Day ...	Monday	October	09	2017	20180521	18045802	-0500
018	Day-After Chris...	Tuesday	December	26	2017	20180521	18045802	-0500
016	Day-After Thank...	Friday	November	24	2017	20180521	18045802	-0500
008	Fathers Day ...	Sunday	June	18	2017	20180521	18045802	-0500
012	Halloween ...	Tuesday	October	31	2017	20180521	18045802	-0500
009	Independence Da...	Tuesday	July	04	2017	20180521	18045802	-0500
010	Labor Day ...	Monday	September	04	2017	20180521	18045802	-0500
002	Martin Luther K...	Monday	January	16	2017	20180521	18045802	-0500
007	Memorial Day ...	Monday	May	29	2017	20180521	18045802	-0500
006	Mothers Day ...	Sunday	May	14	2017	20180521	18045802	-0500
019	New Year's Eve ...	Sunday	December	31	2017	20180521	18045802	-0500
001	New Years Day ...	Sunday	January	01	2017	20180521	18045802	-0500
004	Presidents Day ...	Monday	February	20	2017	20180521	18045802	-0500
005	St Patrick's Da...	Friday	March	17	2017	20180521	18045802	-0500
015	Thanksgiving ...	Thursday	November	23	2017	20180521	18045802	-0500

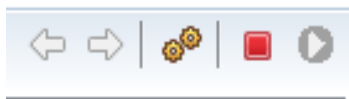
### Record Displayed in Single Record Mode

In the Single-Record Mode, there are only two columns. The field names of the record are listed vertically in the first column, and their corresponding values are listed vertically in the second column. Single-Record display is very useful when you are interested in modifying the data in a single record. It can also be a simpler way to view data in records that have large numbers of fields.








## The File Records Area Toolbar- Read Only Mode

The File Records Area Toolbar for Indexed Files



The File Records Area Toolbar contains the following functions

-  Read Previous (Disabled in Table view mode)
-  Read Next (Disabled in Table view mode)
-  Configure Columns
-  Close File
-  Re-open File

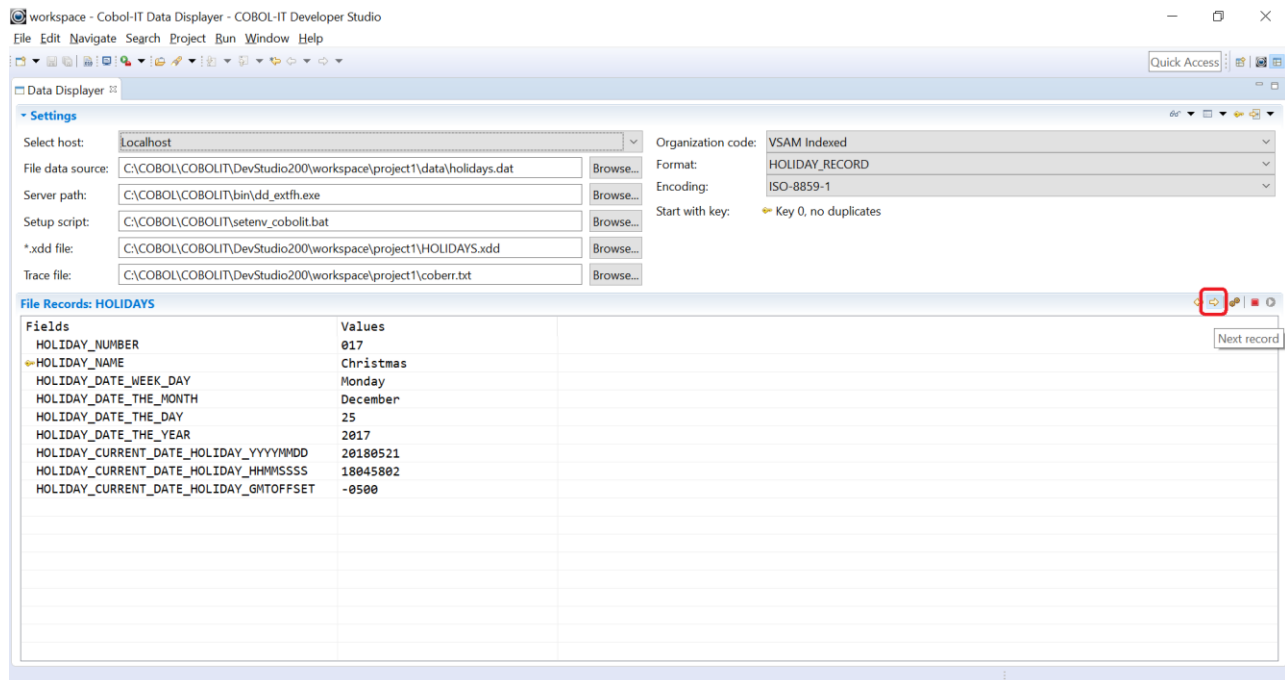
## Read Previous/Read Next

When an Indexed File is open in Single-record Mode, the Read Previous and Read Next functions are enabled. Selecting Read Previous performs a Read Previous operation using the selected Start Key, and displays the Previous Record. Selecting Read Next performs a Read Next operation using the selected Start Key and displays the Next Record.

For the case where one of these operations could not successfully be performed, the operation is disabled. As an example, a Read Previous could not successfully be performed after Opening the File, and doing a First Record read using the selected Start Key. In this case, the Read Previous



button is disabled.



## Configure Columns

When records are being displayed in Table Mode, the field names of the record are stored horizontally in contiguous columns. When there are more column that can fit on a page, a horizontal scroll bar allows you to scroll to the right and reveal the additional columns.

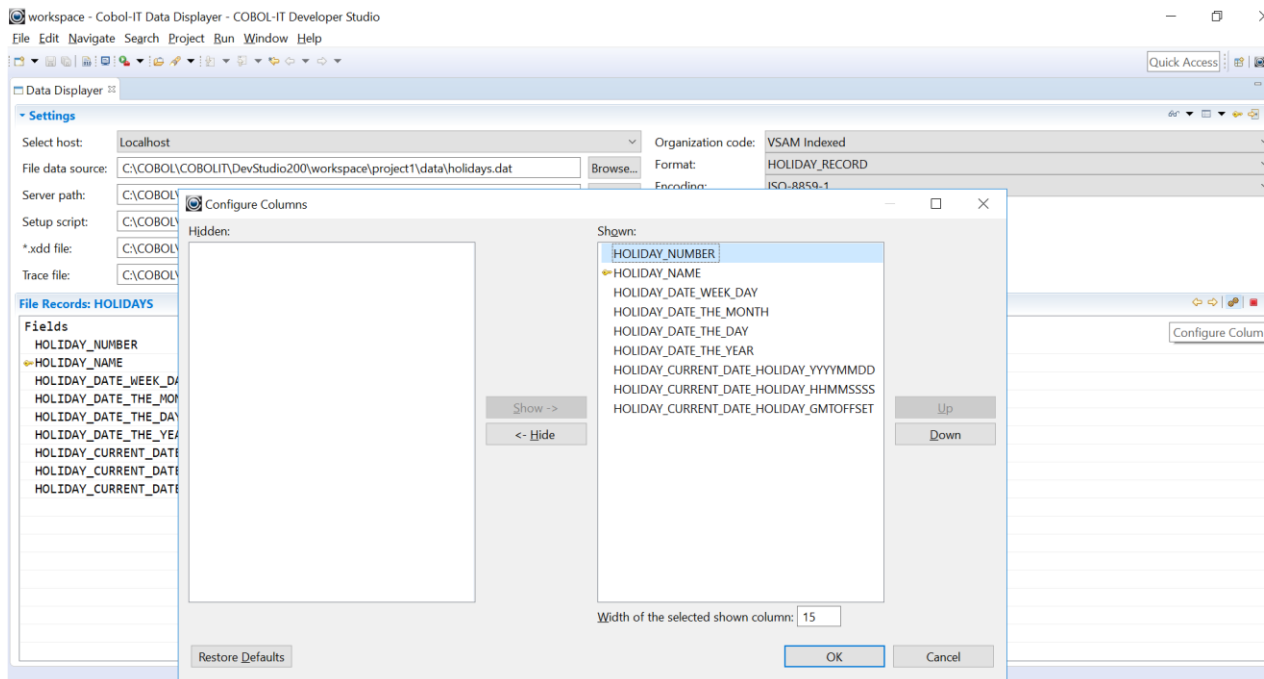
Because of the way data it displayed in Table Mode, it can be useful to use the Configure Columns feature, which allows you reduce the number of columns displayed, by Hiding certain columns, or to change the order in which the columns are displayed, so that the columns that you are interested in browsing and potentially editing appear on the initial display screen.

**To hide a column**, open the Configure Columns interface, select a column name in the “Shown” list, and click on the “Hide” button in the center of the window. The selected column will display in the “Hidden” list, on the left side of the screen. Click “OK” to return to the tabular display, and you will see that the “Hidden” column is no longer displayed.

**To show a column**, open the Configure Columns interface, select a column name in the “Hidden” list, and click on the “Show” button in the center of the window. The selected column will display in the “Shown” list, on the right side of the screen. Click “OK”.

**To set the width of a column**, select a column and enter a number in the “Width of the selected shown column” entry-field. Select “Holiday Current Date Holiday GMT Offset” column and select 15. The Holiday GMT Offset field displays with a width of 15 characters.

**To change the position of the column in the display**, use the “Up” and “Down” on the right side of the screen to change the order in which the columns are displayed. Moving a field “Down” causes it to move to the right in the sequence of columns displayed. Moving a field “Up” causes it to move to the left in the sequence of columns displayed.



## 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 another file. Note that when a file is OPEN in the Data Displayer, it will appear to be OPEN to COBOL programs running in your project. That is, if you leave a file OPEN in the Data Displayer, and return to the Developer Studio, and run a program which OPENS the file, the OPEN will fail. The solution to this problem is to CLOSE the file in the Data Displayer.

## Re-open File

Any time you change the Open mode, the View mode, the Key of reference, or the START function, you must click on the Re-open File button to re-display your data. When switching between READ and READ-WRITE mode, the Re-open File function will first CLOSE, then Re-open the file.

## Using Read/Write Mode

Use the “Open Mode” button on the Data Displayer Toolbar to toggle to Read-Write Mode, and then use the Re-open File function. Re-open File causes the file open in Read-Write Mode to be CLOSED, and then re-opened in Read-Write mode.

In Read-Write mode, you will have access to all of the editing functions of the Data Displayer, which include Add Record, Delete Record, Modify Record (ASCII) and Modify Record (Hex).










## The File Records Area Toolbar- Read-Write Mode

The File Records Area Toolbar for Indexed Files

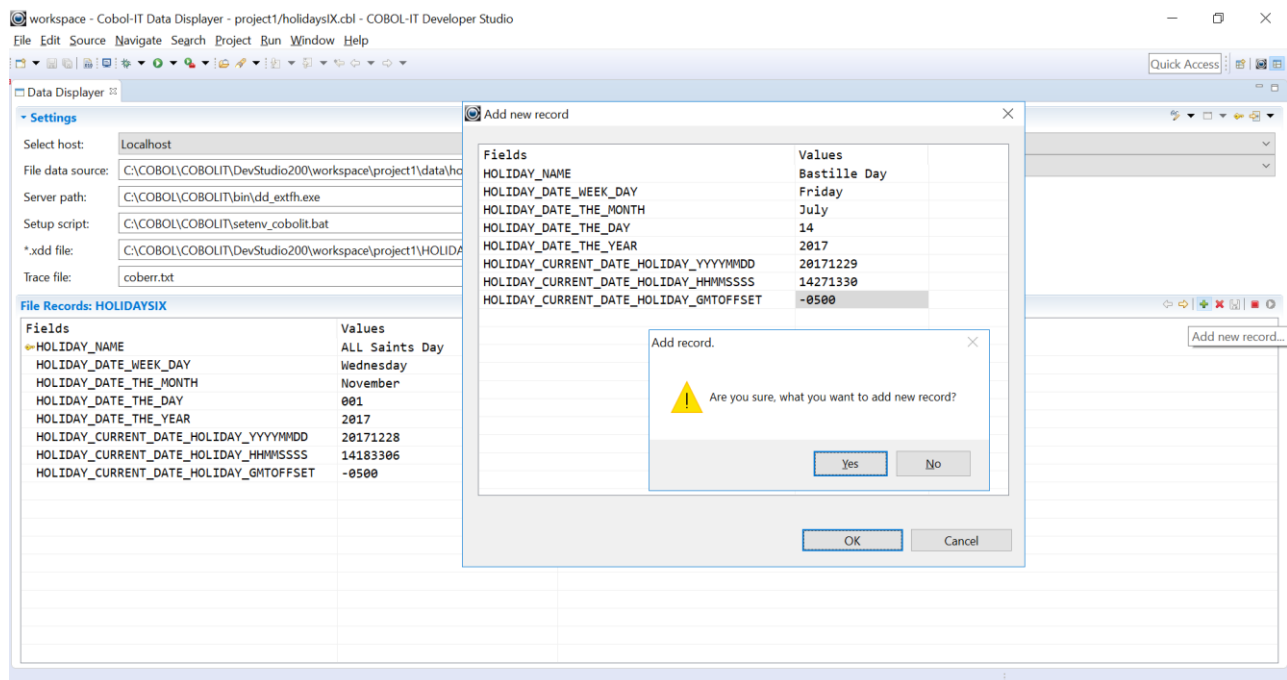


The File Records Area Toolbar contains the following functions

-  Read Previous
-  Read Next
-  Add a Record
-  Delete a Record
-  Save Changes
-  Close File
-  Re-open File

## Add a Record

To add a record, click on the “Add” button on the File Record toolbar. In the Add new record dialog box, the field names are listed in the column on the left. Values are empty in the column on the right. Primary key fields are required. Click “OK”. Verify that you wish to add the record.



## Delete a Record

Select the “Delete” button on the File Records toolbar, and “Yes” on the “Are you sure” dialog box.



workspace - Cobol-IT Data Displayer - COBOL-IT Developer Studio

File Edit Navigate Search Project Run Window Help

Quick Access

Settings

Select host: Localhost Organization code: VSAM Indexed

File data source: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\data\holidays.dat Browse...

Format: HOLIDAY\_RECORD

Server path: C:\COBOL\COBOLIT\bin\dd\_extfh.exe Browse...

Encoding: ISO-8859-1

Setup script: C:\COBOL\COBOLIT\setenv\_cobolit.bat Browse...

Start with key: Key 0, no duplicates

\*.xdd file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\HOLIDAYS.xdd

Trace file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\coberr.txt

Delete record.

Are you sure, what you want to delete this record?

Yes No

Delete record

Fields	Values
HOLIDAY_NUMBER	013
HOLIDAY_NAME	ALL Saints Day
HOLIDAY_DATE_WEEK_DAY	Wednesday
HOLIDAY_DATE_THE_MONTH	November
HOLIDAY_DATE_THE_DAY	01
HOLIDAY_DATE_THE_YEAR	2017
HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD	20180521
HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS	18045802
HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET	-0500

## Modify a Record (ASCII )

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. Unsaved changes are indicated with an asterisk (\*) next to the “Data Displayer” tab title.

workspace - Cobol-IT Data Displayer - COBOL-IT Developer Studio

File Edit Navigate Search Project Run Window Help

Quick Access

Settings

Select host: Localhost Organization code: VSAM Indexed

File data source: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\data\holidays.dat Browse...

Format: HOLIDAY\_RECORD

Server path: C:\COBOL\COBOLIT\bin\dd\_extfh.exe Browse...

Encoding: ISO-8859-1

Setup script: C:\COBOL\COBOLIT\setenv\_cobolit.bat Browse...

Start with key: Key 0, no duplicates

\*.xdd file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\HOLIDAYS.xdd

Trace file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\coberr.txt

File Records: HOLIDAYS

Fields	Values
HOLIDAY_NUMBER	013
HOLIDAY_NAME	ALL Saints Day
HOLIDAY_DATE_WEEK_DAY	Thursday
HOLIDAY_DATE_THE_MONTH	November
HOLIDAY_DATE_THE_DAY	01
HOLIDAY_DATE_THE_YEAR	2017
HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD	20180521
HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS	18045802
HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET	-0500

## Modify a Data Element (Hex)

Single-click on a data element in a record. Right-click and select the “Edit hexadecimal value”







button. In the Set Value (Hex) window, modify the hexadecimal value by overtyping the two-digit number. Unsaved changes are indicated with an asterisk (\*) next to the “Data Displayer” tab title.

The screenshot shows the COBOL-IT Data Displayer window. The 'Settings' tab is active, showing various configuration options. The 'File Records: HOLIDAYS' section displays a table of fields and their values. A 'Set Value (Hex)' dialog is open, allowing the user to modify the hexadecimal value for the selected field.

Fields	Values
HOLIDAY_NUMBER	013
HOLIDAY_NAME	ALL Saints Day
HOLIDAY_DATE_WEEK_DAY	Thursday
HOLIDAY_DATE_THE_MONTH	November
HOLIDAY_DATE_THE_DAY	01
HOLIDAY_DATE_THE_YEAR	2017
HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD	20180521
HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS	18045802
HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET	-0500

The 'Set Value (Hex)' dialog shows a hexadecimal value of 18045802. The 'Hexadecimal' radio button is selected.

## Save Changes

Click on “Save changes” to save your modifications. The field is no longer rendered in bold italics. The asterisk (\*) next to the Data Displayer title has been cleared.

The screenshot shows the COBOL-IT Data Displayer window after saving changes. The 'File Records: HOLIDAYS' section displays the same table of fields and values. The 'Set Value (Hex)' dialog is closed, and the 'Save changes...' button is visible in the top right corner.

Fields	Values
HOLIDAY_NUMBER	013
HOLIDAY_NAME	ALL Saints Day
HOLIDAY_DATE_WEEK_DAY	Thursday
HOLIDAY_DATE_THE_MONTH	November
HOLIDAY_DATE_THE_DAY	01
HOLIDAY_DATE_THE_YEAR	2017
HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD	20180521
HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS	18055802
HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET	-0500

The 'Save changes...' button is visible in the top right corner.







### Close File

If you try to close a file with unsaved changes, you will be asked if you wish to save your changes.

workspace - Cobol-IT Data Displayer - COBOL-IT Developer Studio

File Edit Navigate Search Project Run Window Help

Quick Access

\*Data Displayer

Settings

Select host: Localhost Organization code: VSAM Indexed

File data source: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\data\holidays.dat Browse... Format: HOLIDAY\_RECORD

Server path: C:\COBOL\COBOLIT\bin\dd\_extfh.exe Browse... Encoding: ISO-8859-1

Setup script: C:\COBOL\COBOLIT\setenv\_cobolit.bat Browse... Start with key: Key 0, no duplicates

\*.xdd file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\HOLIDAYS.xdd Browse...

Trace file: C:\COBOL\COBOLIT\DevStudio200\workspace\project1\coberr.txt Browse...

File Records: HOLIDAYS

Fields	Values
HOLIDAY_NUMBER	013
HOLIDAY_NAME	ALL Saints Day
HOLIDAY_DATE_WEEK_DAY	Thursday
HOLIDAY_DATE_THE_MONTH	November
HOLIDAY_DATE_THE_DAY	01
HOLIDAY_DATE_THE_YEAR	2017
HOLIDAY_CURRENT_DATE_HOLIDAY_YYYYMMDD	20180521
HOLIDAY_CURRENT_DATE_HOLIDAY_HHMMSSSS	18045802
HOLIDAY_CURRENT_DATE_HOLIDAY_GMTOFFSET	-0500

Edit record.

You have unsaved changes. Please indicate how you would like to proceed.

Save and close Return to File Editor Don't save and close





# Code Coverage

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

After you have set your Project>Properties to include the code coverage compiler flags, code coverage analysis is produced automatically when you run ( or run in debug ) your program. The results are a Coverage analysis are 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.

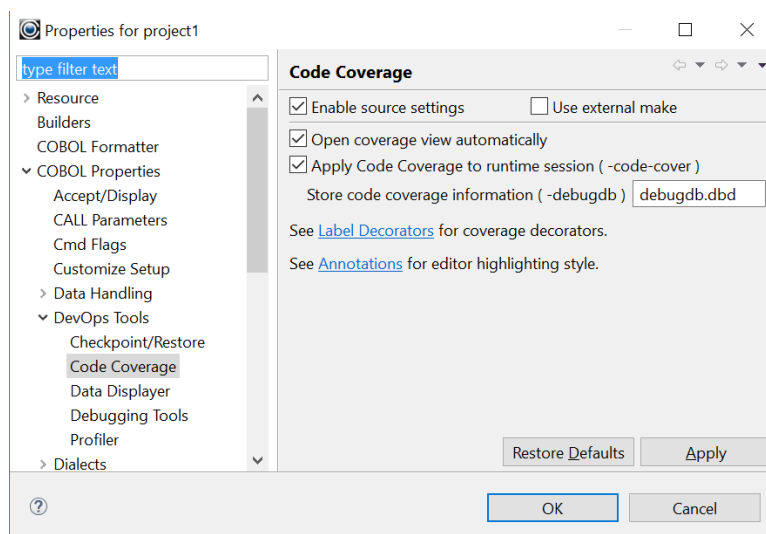
## Quick Start Guide

### Enable COBOL code coverage in Project>Properties

The behavior of the COBOL Code Coverage can be adjusted in the Project>Properties>COBOL Properties>DevOps Tools>Code Coverage dialog screen.

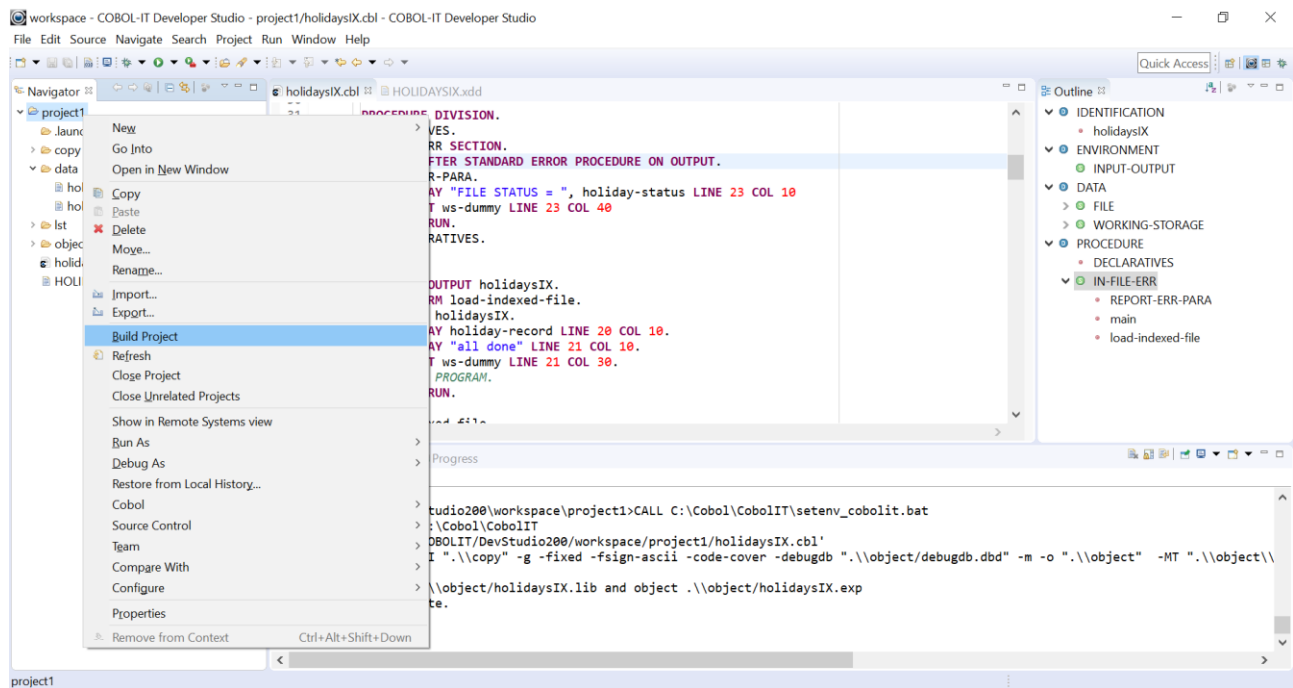
To launch coverage for a COBOL program, you must build the project using the `-code-cover` compiler flag, with the `-debugdb` compiler flag, followed by a the name of the `.dbd` file in which you wish to store your coverage information.

During the project build, the compiler will prepare all required metadata for coverage and store it in the `debugdb.dbd` file. This file is generated in the same directory as the object files. If no `COB_LIBRARY_PATH` is set, the `.dbd` file and the object files are generated in the current working directory. Otherwise the `.dbd` file and the object files are generated in the directory named by the `COB_LIBRARY_PATH` environment variable.

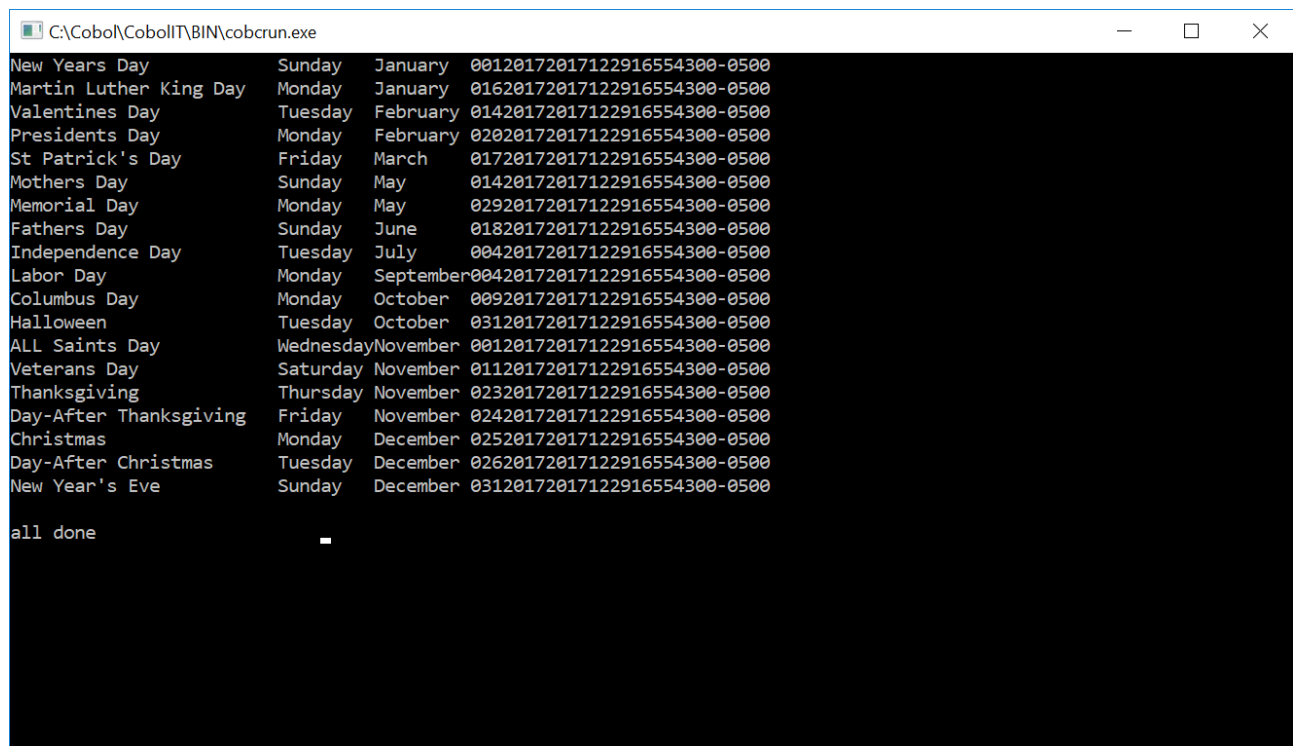




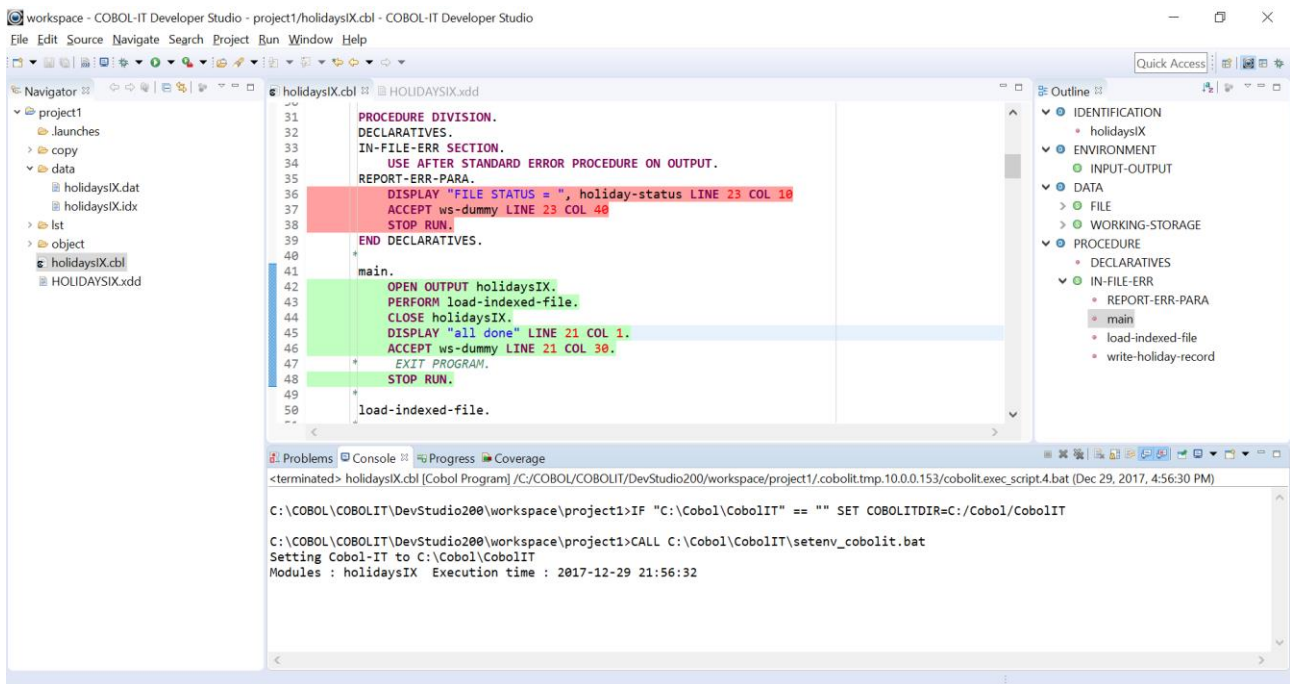
## Build the Project



## Run the program



## View results



## Overview

COBOL Code Coverage is based on [EclEmma](#) project. EclEmma is a free Java code coverage tool for Eclipse, available under the [Eclipse Public License](#). The documentation is an adapted version of [EclEmma User Guide](#).

COBOL Code Coverage records which parts of your COBOL code are executed during a particular program launch. Therefore coverage analysis always involves two steps:

1. Run the program
2. Analyze coverage data

[Running](#) a coverage analysis is as simple as pressing a single button like the existing Run and Debug buttons. The coverage results are automatically summarized in the [Coverage view](#) and highlighted in the [COBOL editors](#).



## Launching in Coverage Mode

### Source Code Annotations

Line coverage of the active coverage [session](#) is directly displayed in the COBOL source editors.

The screenshot shows the COBOL-IT Developer Studio interface. The main editor displays the source code of 'load-file.cpy' with line coverage annotations. Lines 36-38 are highlighted in red, indicating they have not been executed. Lines 42-47 are highlighted in green, indicating they are fully covered. The 'load-indexed-file' procedure is also highlighted in green. The 'Outline' pane on the right shows the project structure. The 'Coverage' pane at the bottom displays a table with the following data:

Element	Coverage	Covered Lin...	Missed Lines	Total Lines
project1	98.4 %	179	3	182
holidaysIX.cbl	66.7 %	6	3	9
copy	100.0 %	173	0	173
load-file.cpy	100.0 %	173	0	173

Source lines that have been covered, and source code lines that have been missed are highlighted according to the following color code:

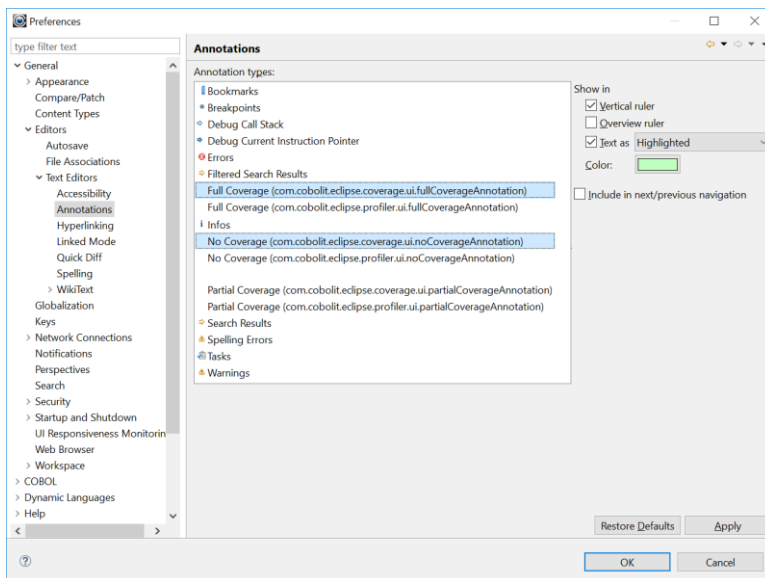
- green for fully covered lines
- red for lines that have not been executed

Source annotations automatically disappear when you start editing a source file or delete the coverage session.

Colors for Source annotations can be modified in the Window>Preferences dialog. To modify the color selections for Source Annotations, go to :

Window>Preferences>General>Appearance>Editors>Text Editors>Annotations.





The corresponding entries are :

Full Coverage

No Coverage

## Decorators

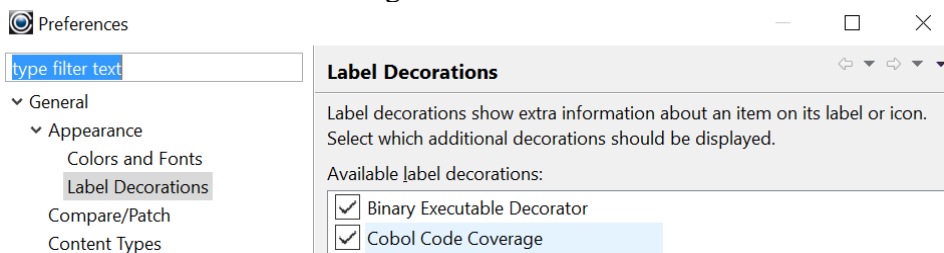
**Note:** Decorators are not enabled by default.

Decorators add graphical and textual information to elements shown in workbench views. COBOL Code Coverage provides coverage decorators for the source files listed in the Elements column of the Coverage View and in the Navigator Window that are being used in the currently active [coverage session](#).

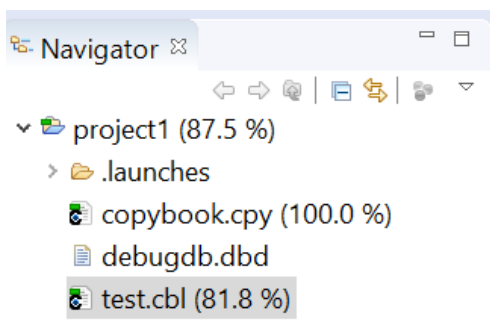
Code Coverage decorators include small green/red bars which appear next to the source files listed in the Elements column of the Coverage View, and a percentage value which appears next to the names of the source files listed in the Navigator Window. .



To set Decorators for the Navigator Window, go to:  
**Window>Preferences>General>Appearance>Label Decorations.**  
Select the Cobol Code Coverage checkbox.



The percentage shown is calculated based upon the covered lines. Coverage decorators are only visible if there is an active coverage session and only shown for elements containing executable code. As an example, see the image below :

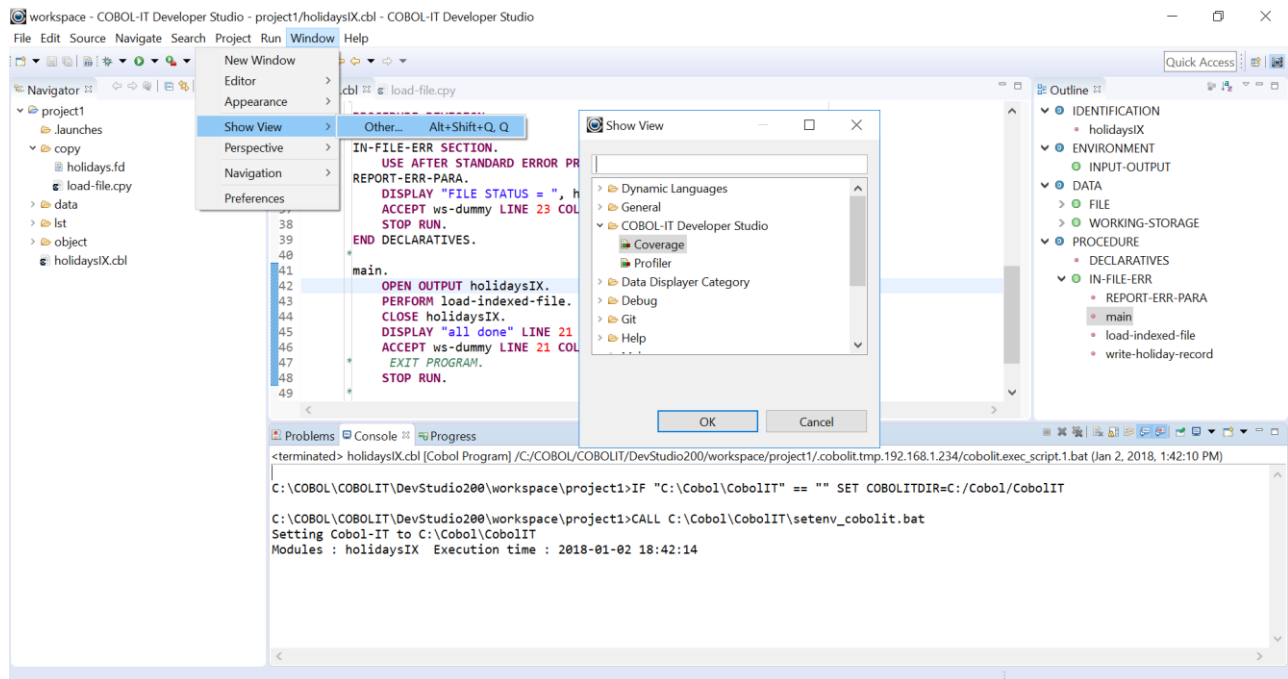




## The Coverage View

### Opening the Coverage View

By default, the Coverage View opens when a new [coverage session](#) is added. As with other Views, it can be manually opened from the Window>Show View menu.



### Coverage View Summaries

The Coverage View shows coverage summaries for the Active Session.

Element	Covera...	Covered Lin...	Missed Lines	Total Lines
project1	98.4 %	179	3	182
holidaysIX.cbl	66.7 %	6	3	9
copy	100.0 %	173	0	173
load-file.cpy	100.0 %	173	0	173

The Coverage View opens with the summary totals for the project.

Click on the dropdown arrow to the left of the project name to see the summary information displayed for each of the source files. Source files listed include copy files containing source code from within the programs that were run.

In the Coverage View, you will find the following columns:

Element      Name of Source file. Clicking on the source file in the Element column opens the







file in the Editor, which will contain highlighted source code, where Covered Lines and Misses Lines are highlighted in different colors. Default colorizations for the highlights are green to highlight covered lines and red to highlight missed lines.

Coverage	Reports % of lines covered
Covered line	Count of the lines executed in the source file
Missed lines	Count of the lines not executed in the source file
Total Lines	Sum of Covered Lines and Missed Lines

The elements may be sorted in ascending or descending order by clicking the respective column header.

## The Coverage View Toolbar



Relaunch Coverage Session

Re-run the currently selected coverage session.



Remove Active Session

Remove the currently selected coverage session.



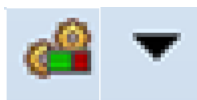
Remove All Sessions

Remove all coverage sessions.



Merge Sessions

Merges multiple sessions into a single session.



Select Active Session

Select [session](#) from the drop down-menu and make it the active session. Each active session includes the cumulative results of tests in that session.



Collapse All

Collapse all expanded tree nodes. In a collapsed tree node, only the project-level statistics are visible.



Link with Current Selection

Causes the coverage view element selected to reveal the COBOL source currently selected in other views or editors. As an example, when Link with Current Selection is selected, you can click on “hello.cbl” in the Element list to open “hello.cbl” in the Code Editor.



Show Menu

Allows selection of “Hide Unused Elements”.

Minimize

Minimizes the Coverage View

Maximize

Maximizes the Coverage View

## Managing Coverage Sessions

A coverage session is the code coverage information of particular program run. It contains the list of COBOL Elements, with the coverage information recorded for each.

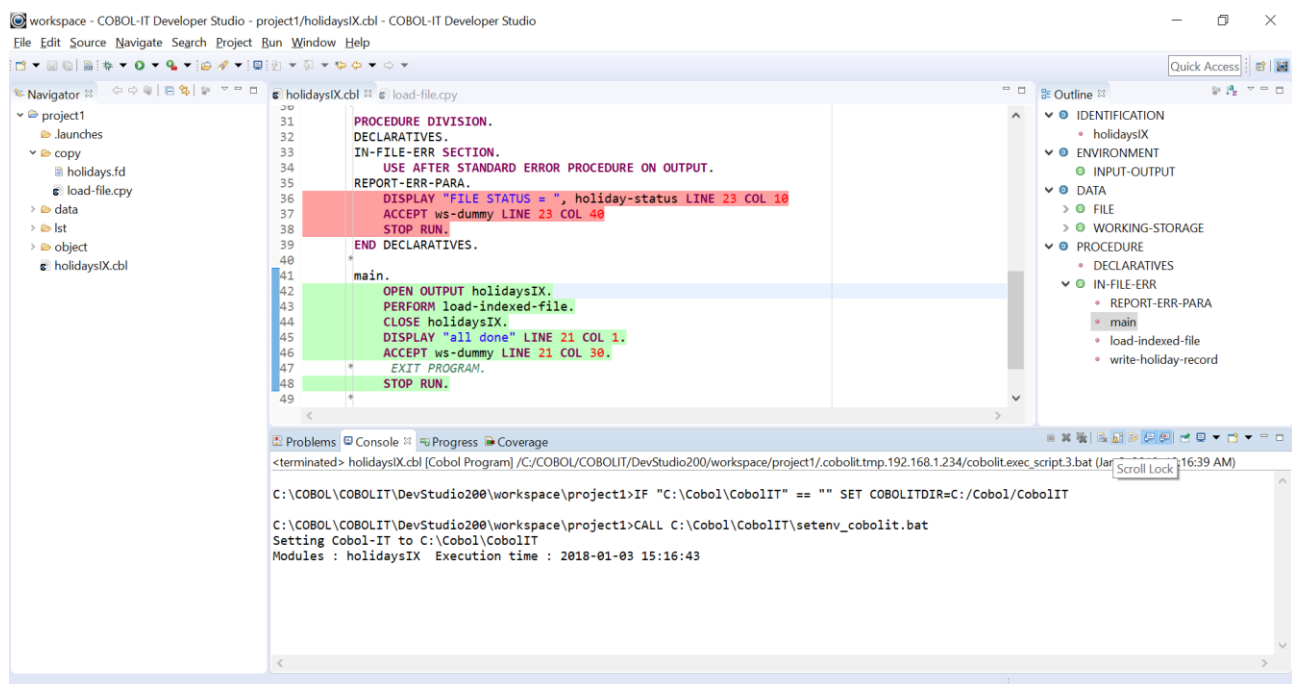
### Session Lifecycle

A coverage session is automatically created at the end of each [coverage launch](#). Alternatively, sessions can be [imported](#) from external launches. The [coverage view](#) allows removing sessions.

All coverage sessions are deleted when the workbench is closed.

### Relaunch Coverage Session

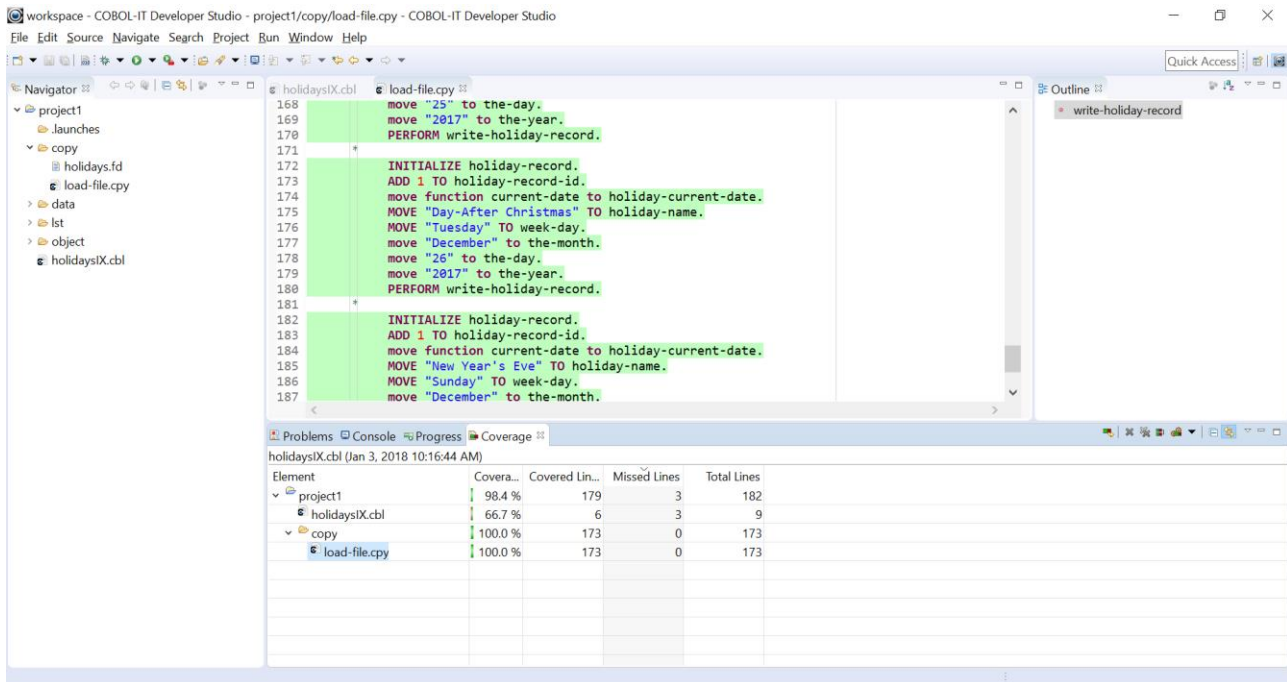
The Relaunch Coverage Session toolbar button causes the application to be re-launched. When the application session has completed the Coverage results will be displayed as the new “Active Session”.





## Link with Current Selection

The Link with Current Selection toolbar button opens the selected source file in the Code Editor window.



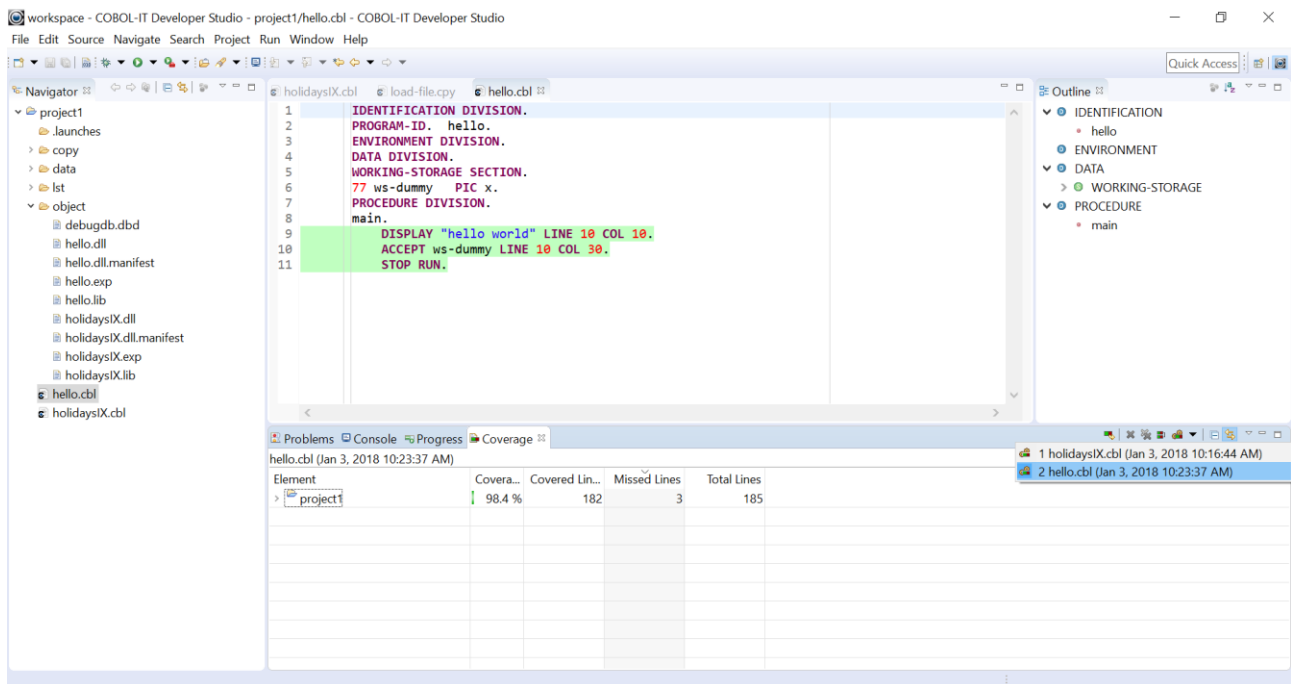
## The Active Session

Even if there can be multiple coverage sessions, only one session can be the active coverage session. The active session can be selected from a drop-down list in the [coverage view](#) and defines the input of this view as well as the [COBOL source highlighting](#).

## Select Active Session

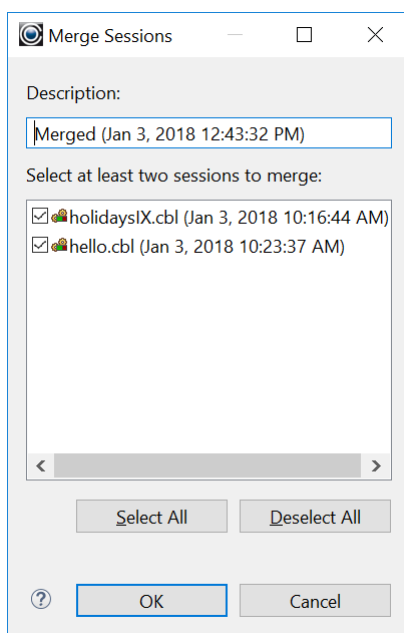
The Active Session dropdown box displays the currently active session and sessions which can be selected as the Active Session. The selected session is displayed in the Coverage View.





## Merging Sessions

If the overall test set consists of multiple test launches, they will result in multiple different coverage sessions. For analysis it may make sense to combine these sessions to a single session. If there is more than one session the [coverage view](#) provides the Merge Sessions command. This command allows selecting a subset from the existing sessions and combining it to a single coverage session.



## Session Import and Export

COBOL Code Coverage provides import and export functionality.

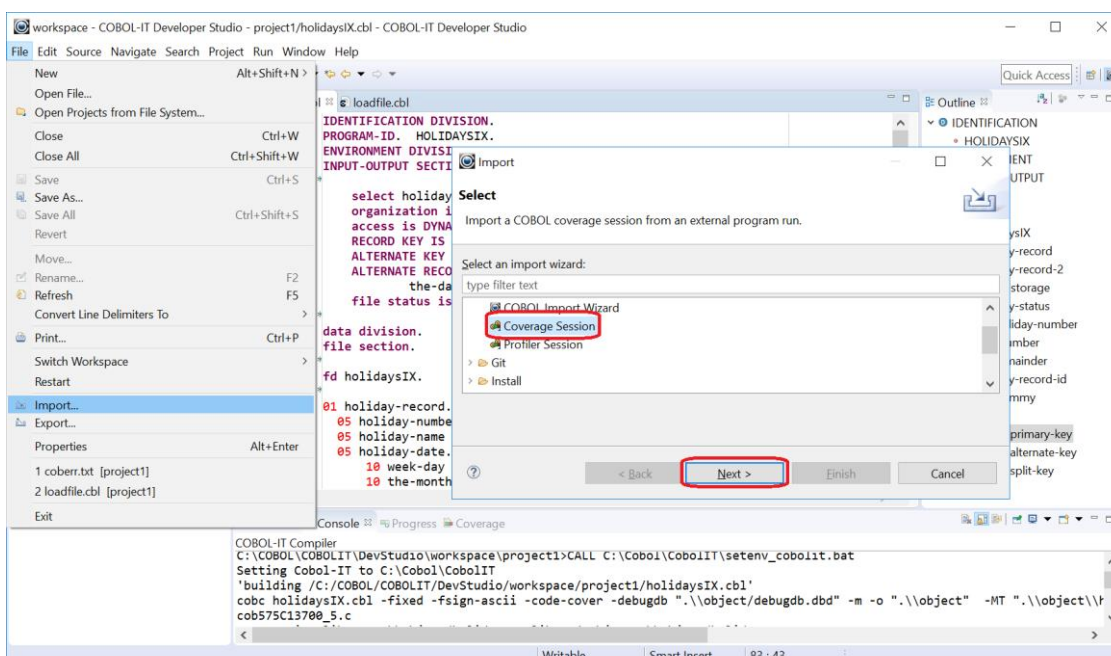
### Coverage Session Import

If your program is launched outside the COBOL-IT Developer Studio, you might import execution data from these launches. This data could then be merged with data captured in sessions in the Developer Studio. This allows to study the coverage results directly in your source code. The Coverage Session import wizard can be activated from the File → Import... menu or from the Coverage view's context menu.

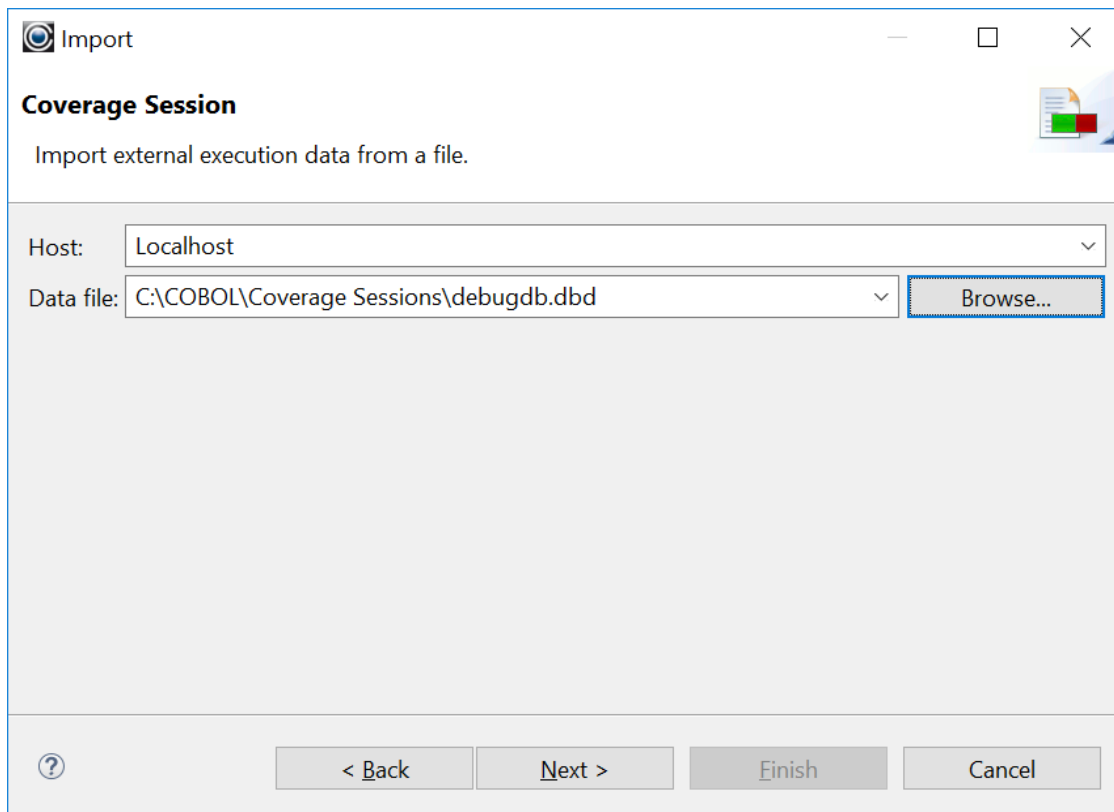
The wizard dialog requires you to specify the source of the execution data on its first page. On the second page you can specify session name and the scope, which are the projects with COBOL sources that should be considered.

**Warning:** Imported execution data must be based on the exact same COBOL source files that are also used within the COBOL-IT Developer Studio. If the external launch was based on different COBOL sources coverage will be shown incorrectly.

To import a Code Coverage session, select the File>Import function from the Main Menubar. On the subsequent Import screen, select “Coverage Session” from among the available COBOL import wizards.



Click on the "Next" button to open a dialog Window, and select the Coverage Session you wish to import. Code Coverage sessions are stored in .dbd files, as described earlier. In our example below, we have used the Browse button to navigate to a .dbd file in which we have saved information.



**Import**

**Coverage Session**

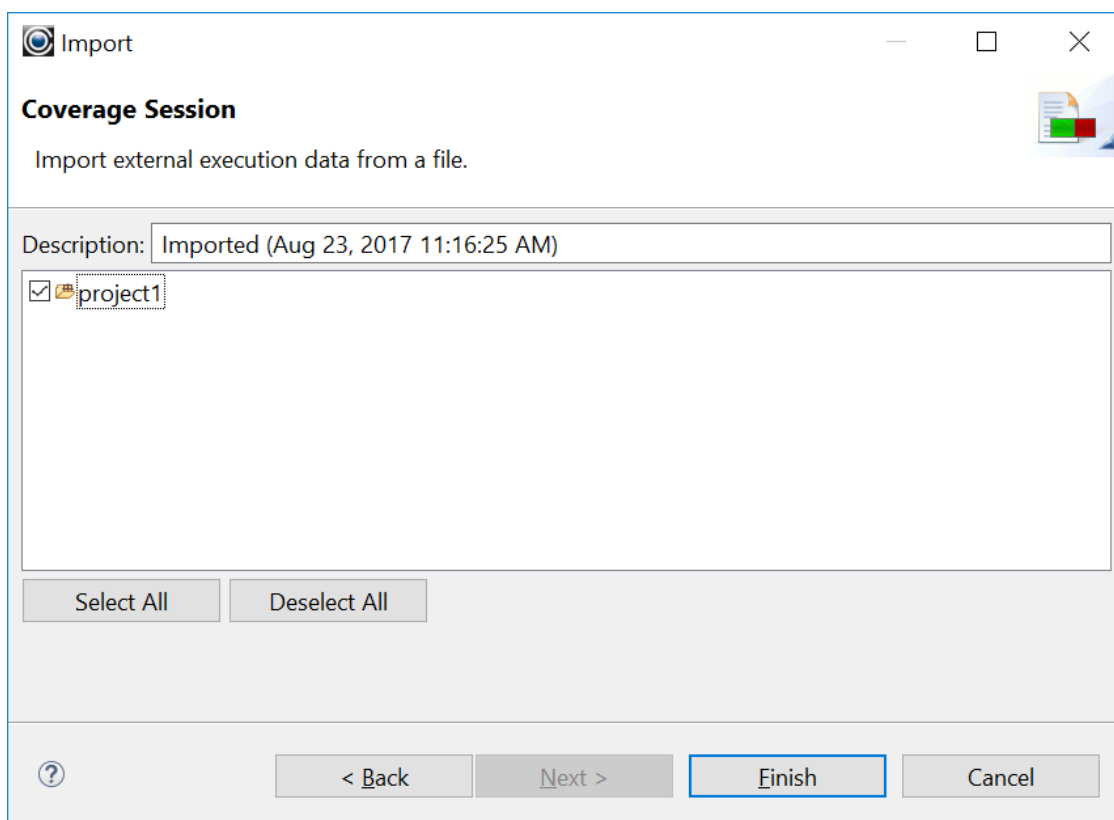
Import external execution data from a file.

Host: Localhost

Data file: C:\COBOL\Coverage Sessions\debugdb.dbd Browse...

? < Back Next > Finish Cancel

Click “Next” to advance to the final screen, where you can select a specific session, or indicate the folders that are used in the project.




**Import**

**Coverage Session**

Import external execution data from a file.

Description: Imported (Aug 23, 2017 11:16:25 AM)

☒  project1

Select All Deselect All

? < Back Next > Finish Cancel

Click “Finish” to import the data from the Coverage Session.

This data will be displayed in the Coverage View, with a notation that the session was imported, and when it was imported.

Imported (Aug 23, 2017 11:16:25 AM)				
Element	Covera...	Covered Lin...	Missed Lines	Total Lines
project1	99.2 %	255	2	257
holidaysIX.cbl	95.5 %	42	2	44
loadfile.cbl	100.0 %	213	0	213

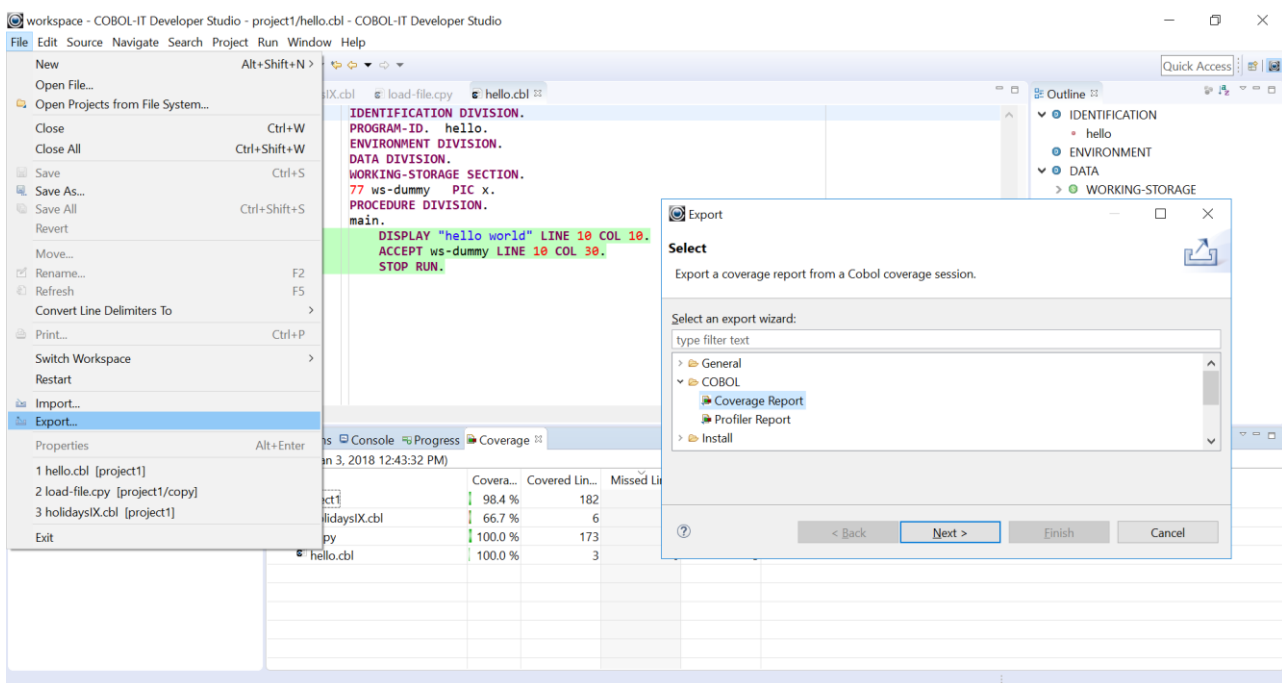
## Coverage Session Export

The session export wizard allows you to export coverage sessions in one of these formats:

- **HTML:** A detailed and browseable report as a set of HTML files.
- **Zipped HTML:** Same as above but zipped into a single file.

To export a Code Coverage session, select the File>Export function from the Main Menubar.

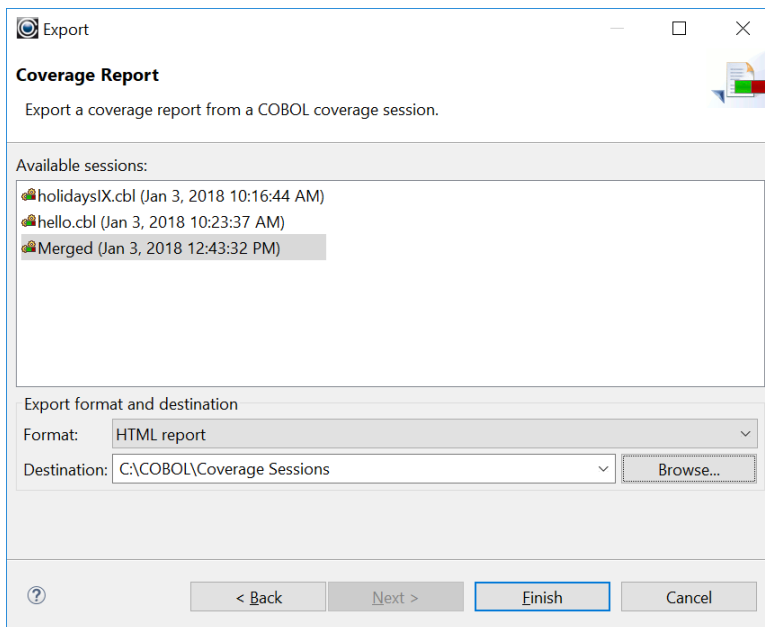
There must be at least one coverage session available to use the export wizard.



Select one of the existing sessions and the export format from the subsequent screen.

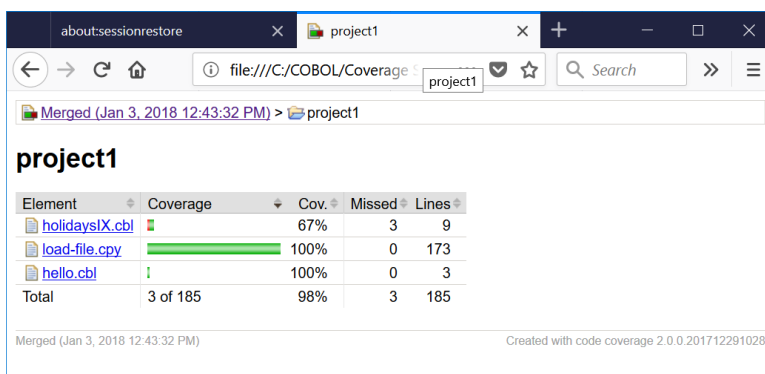
Use the Browse Button to select a destination folder in which to store your exported session file.

Then, click on the “Finish” button.



## Viewing your exported session

To view your exported session, navigate to the folder in which it is saved. Click on the index.html file to open it in your browser. From your browser, you can click on the project name to expand it, and see the individual resources.



You can then click on the resource, to view the annotated source, in which covered and missing lines of code are highlighted in the green/red highlights.





```
19.      WORKING-STORAGE SECTION.  
20.      *  
21.      01 numeral-numeric PIC 999 VALUE 0.  
22.      01 numeral-alpha REDEFINES numeral-numeric PIC XXX.  
23.      77 num-records PIC 999 VALUE 0.  
24.      77 ws-quotient PIC 999.  
25.      77 ws-remainder PIC 9.  
26.      *  
27.      77 holiday-record-id PIC 999 VALUE 1.  
28.      77 ws-dummy pic x.  
29.      77 holiday-status pic xx.  
30.      *  
31.      PROCEDURE DIVISION.  
32.      DECLARATIVES.  
33.      IN-FILE-ERR SECTION.  
34.      USE AFTER STANDARD ERROR PROCEDURE ON OUTPUT.  
35.      REPORT-ERR-PARA.  
36.      DISPLAY "FILE STATUS = ", holiday-status LINE 23 COL 10  
37.      ACCEPT ws-dummy LINE 23 COL 40  
38.      STOP RUN.  
39.      END DECLARATIVES.  
40.      *  
41.      main.  
42.      OPEN OUTPUT holidaysIX.  
43.      PERFORM load-indexed-file.  
44.      CLOSE holidaysIX.  
45.      DISPLAY "all done" LINE 21 COL 1.  
46.      ACCEPT ws-dummy LINE 21 COL 30.  
47.      * EXIT PROGRAM.  
48.      STOP RUN.  
49.      *  
50.      load-indexed-file.  
51.      *  
52.      COPY "load-file.cpy".  
53.
```

## Profiler

The Profiler records where your application uses CPU time, elapsed time, memory, and CPU processing power.

Gathering Profiling data is very simple:

- Compile your source files with `-fprofiling`
- Run the program
- Analyze the coverage data

After you have set your Project>Properties to include the profiling compiler flag, profiling data is produced automatically when you run ( or run in debug ) your program. The results are presented in the Profiler View. Information gathered on the level of the Paragraph/Section included number of times entered, time elapsed, CPU time elapsed, external calls, and time elapsed in external calls. When running in the UNIX/Linux operating environments, you also have access to Memory and CPU usage, displayed in real time.

The Profiler stores the output for a runtime session in a single .xls file that is output at the program exit. The single output file is created using the naming convention:  
`cob_profiling_<PID>_final.xls`.

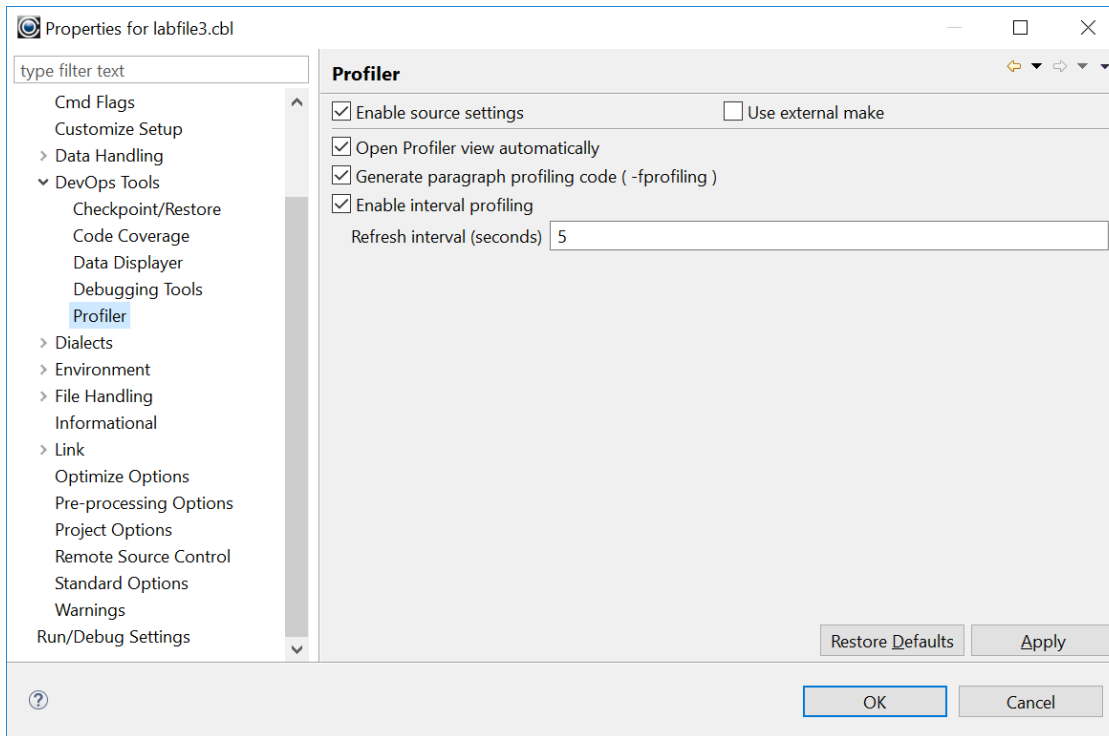
## Profiling Preferences

The behavior of the COBOL Profiling can be adjusted in the Project>Properties>DevOps





Tools>Profiler.



- Open Profiling View automatically: Whenever a new profiling session becomes active the Profiling View can be automatically shown in the current workbench window. (Default: on)
- Generate paragraph profiling code ( -fprofiling ). Sets the -fprofiling compiler flag.

Enable interval profiling>Refresh interval (seconds)- Sets the interval for the profiling data display in the runtime tab of the Profiler View.

## Runtime Environment Variables

### COB\_PROFILING\_DIR

When compiling with the -fprofiling compiler flag, the runtime will check for the COB\_PROFILING\_DIR environment variable, and generate the profiling data file in that directory if it is defined. Otherwise, the profiling data file is created in the same directory as the source file.

Note that COB\_PROFILING\_DIR environment variable requires a trailing slash.

As an example, in Linux/UNIX environments:

```
>export COB_PROFILING_DIR=mydir/
```

As a result, the output file is generated as mydir/cob\_profiling\_<pid>\_final.xls

In Windows environments:

```
>set COB_PROFILING_DIR=mydir\
```

As a result, the output file is generated as mydir\cob\_profiling\_<pid>\_final.xls



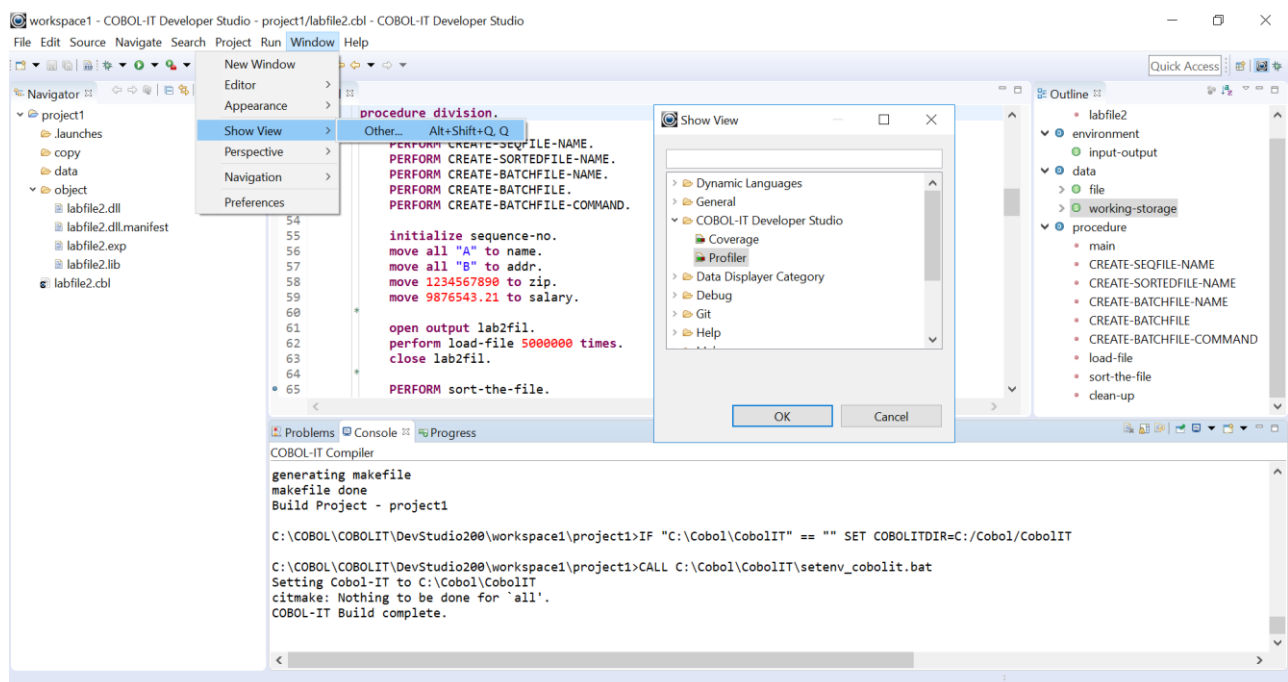
## COB\_PROFILING\_EACH\_MODULE

The COB\_PROFILING\_EACH\_MODULE runtime environment variable, when set to Y, causes the profiler to revert to the old profiling behavior, in which the .xls file is output at the exit of each module in an application.

## The Profiler View

### Opening the Profiler View

The Profiler view is opened automatically after a program is run in profiling mode. As with other Views, it can be manually opened from the Window>Show View menu.



The Profiler view contains two tabs: Runtime and Paragraphs. The Runtime tab is available in Linux/UNIX operating environments and contains information about current cpu and memory usage of the running program. The Paragraphs tab is empty during execution. On program exit, the tab is activated and filled with the profiling information that has been collected in the cob\_profiling\_[pid]\_final.xls file.



## Paragraphs Tab

The Paragraphs Tab of the Profiler View shows profiling information collected during the runtime session.

### Profiler View Summaries

Paragraph	Entry count	Total CPU	External call ...	Total Elaps	External call ...	CPU Intrinsic	CPU Intrinsic...	CPU Total a...	Elaps Intrinsic	Elaps Intrinsic...	Elaps Total ...
labfile2: MAIN SECTION.label_sort-	1	384	384	384000	384000	0	0	384	0	0	384000
labfile2: MAIN SECTION.label_clean	1	147	147	147000	147000	0	0	147	0	0	147000
labfile2: labfile2	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.Default Err	0	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_CREA	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_CREA	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_CREA	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_CREA	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_CREA	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_load-	10	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.label_main	1	0	0	0	0	0	0	0	0	0	0
labfile2: MAIN SECTION.MAIN SEC	1	0	0	0	0	0	0	0	0	0	0

On the Title bar, you see the name of the main program, and the data-time stamp for the profiling session.

In the Paragraphs Tab, you will find the following columns:

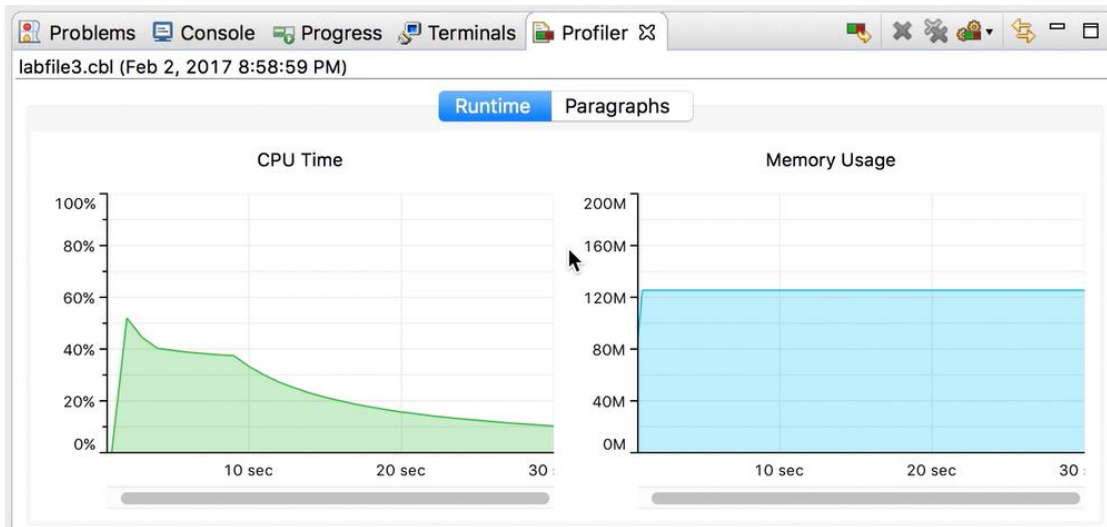
Paragraph	In the form [program name]:[section name].label_[paragraph name]
Entry Count	The number of times the paragraph was entered
Total CPU	CPU time in thousandths of a second
External CALL CPU	CPU time spent in external calls in thousandths of a second
Total Elaps	Total elapsed time in millionths of a second
External Call Elaps	Total elapsed time in external calls in millionths of a second
CPU Intrinsic	CPU time executing Intrinsic Functions
CPU Intrinsic avg run	CPU time executing Intrinsic Functions (average run)
CPU Total avg run	CPU Total (average run)
Elaps Intrinsic	Elapsed time executing Intrinsic Functions
Elaps Intrinsic avg run	Elapsed time executing Intrinsic Functions (average run)
Elaps Total avg run	Total elapsed time (average run)

The elements may be sorted in ascending or descending order by clicking the respective column header.



## Runtime Tab

In Linux/UNIX operating environments, the Runtime tab contains two graphics with cpu and memory information of the running program. During program execution graphics data is updated at an interval rate that is configurable by the user.



## The Profiler View Toolbar



Relaunch Profiler Session

Re-run the currently selected profiler session.



Remove Active Session

Remove the currently selected profiler session.



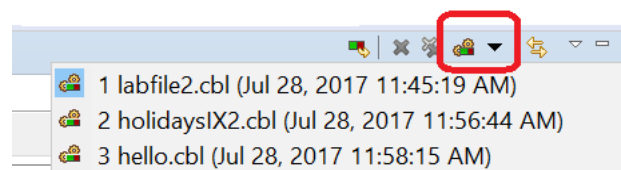
Remove All Sessions

Remove all profiler sessions.



Select Active Session

Select [session](#) from the drop down-menu and make it the active session.



Link with Current Selection

When this toggle is checked the Profiler View automatically opens target source code when you select a paragraph in the Paragraphs tab.



Minimize

Minimizes the Coverage View



Maximize

Maximizes the Coverage View

## Relaunch Profiler Session

The Relaunch Profiler Session toolbar button causes the application to be re-launched in the debugger. When the application session has completed the Profiling results will be displayed as the new “Active Session”.

## Select Active Session

The Active Session dropdown box displays the currently active session and sessions which can be selected as the Active Session. The selected session is displayed in the Profiler View.

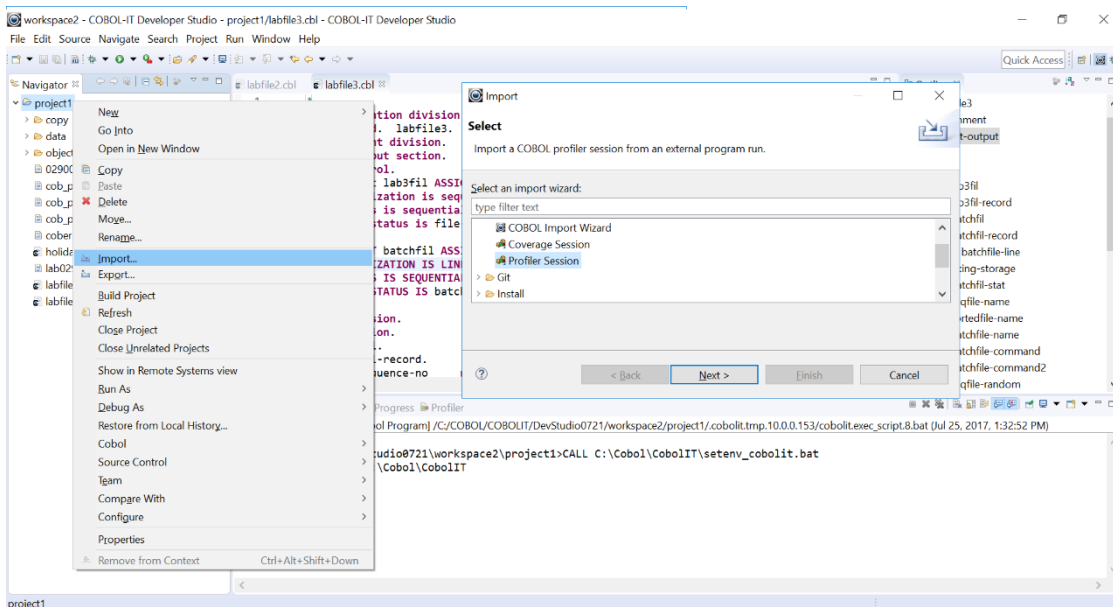
## Link with Current Selection

The Link with Current Selection toolbar button opens the selected source file in the Code Editor window.

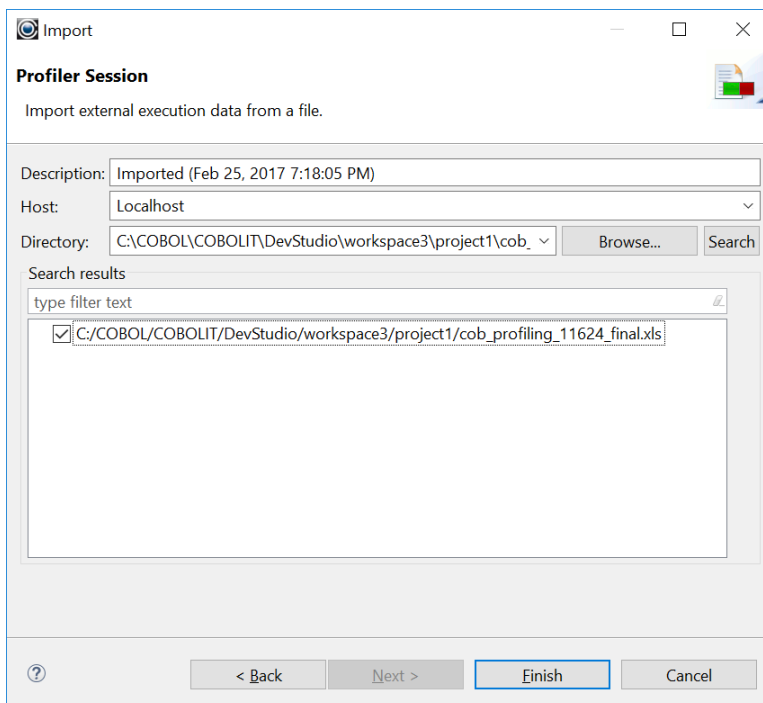
## Profiler Session Import

Profiling sessions can be launched from the COBOL command line. As we have seen, when you compile a program with the -fprofiling compiler flag, and then run the program, the runtime will create an output file called cob\_profiling\_[pid]\_final.xls. This file can be imported into the Developer Studio.

To import a Code Coverage session, select the File>Import function from the Main Menubar. On the subsequent Import screen, select “Profiler Session” from among the available COBOL import wizards.



Click on the "Next" button to open a dialog Window, and select the Profiler Session you wish to import. Profiler sessions are stored in .xls files, as described earlier. In our example below, we have used the Browse button to navigate to a .xls file in which we have saved information.



Click "Finish" to import the data from the Profiler Session. This data will be displayed in the Paragraphs Tab of the Profier View, with a notation that the session was imported, and when it was imported.

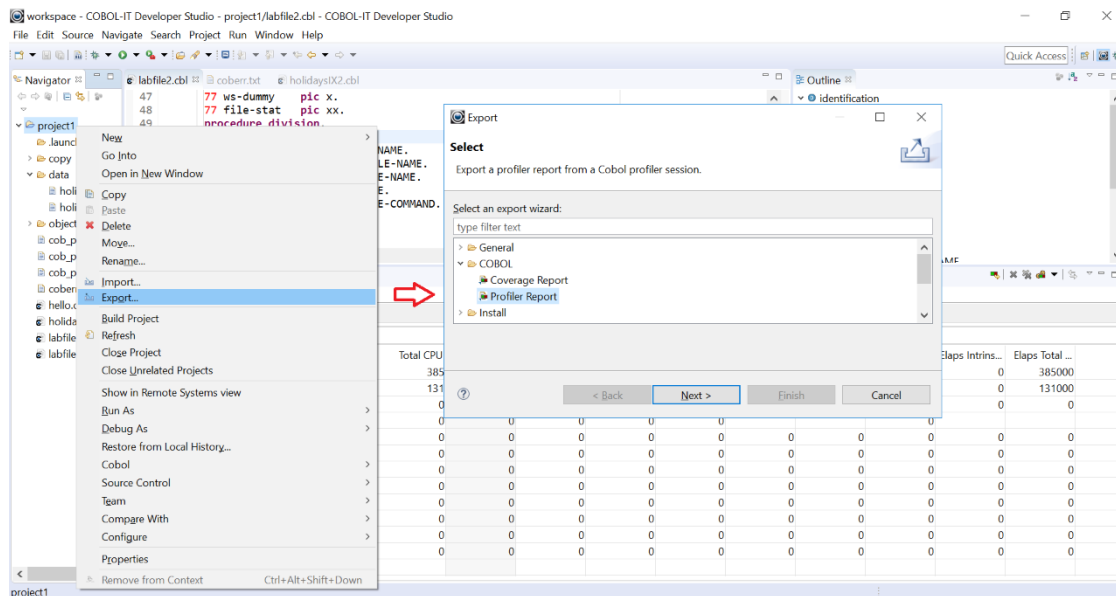


Paragraph	Entry count	Total CPU	External call CPU	Total Elaps	External call E
labfile2: MAIN SECTION.Label_main	1	18,731	18,214	18,758,000	18,214
labfile2: MAIN SECTION.Label_load	2,000,000	1,686	0	1,659,000	
labfile2: MAIN SECTION.Label_CRE	1	14	0	14,000	
labfile2: MAIN SECTION.Default Err	0	0	0	0	
labfile2: MAIN SECTION.Label_CRE	1	0	0	0	
labfile2: labfile2	1	0	0	0	
labfile2: MAIN SECTION.Label_CRE	1	0	0	0	
labfile2: MAIN SECTION.Label_CRE	1	0	0	0	
labfile2: MAIN SECTION.Label_CRE	1	0	0	0	
labfile2: MAIN SECTION.Label_CRE	1	0	0	0	
labfile2: MAIN SECTION.MAIN SEC	1	0	0	0	

## Profiler Session Export

Profiling sessions that have been generated from within the Developer Studio can be exported to a location of your choice.

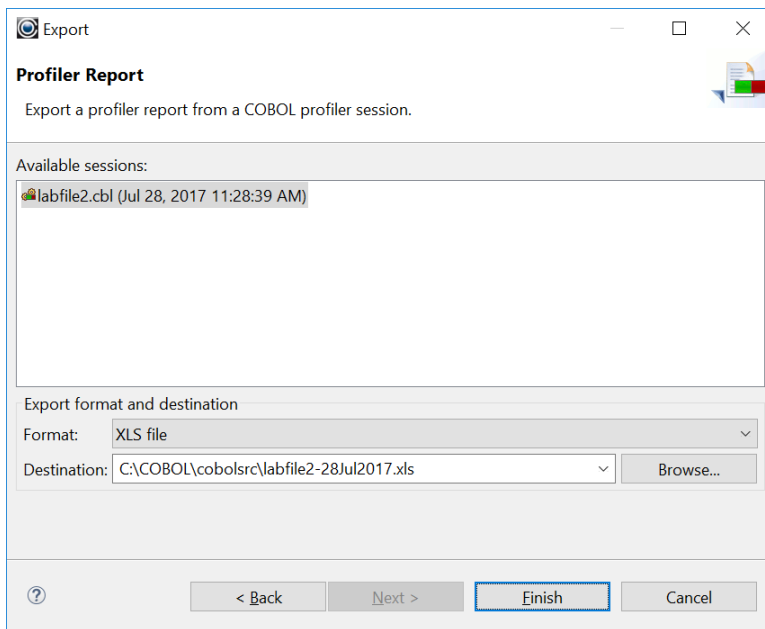
To export a Code Coverage session, select the File>Export function from the Main Menubar. On the subsequent Export screen, select “Profiler Report” from among the available COBOL export wizards.



Click on the “Next” button to open the “Export” dialog window. Then click on “Finish” to export the profiler report to the Export destination.







•

## Using Git for Source Code Control

**Git** is a source code control system that was first developed by Linus Torvalds in 2005 for the development of the Linux kernel. Like Linux, Git is free software, distributed under the terms of the GNU GPL 2. Git provides the ability to create a source code repository locally, or to access an existing source code repository hosted on the web. COBOL-IT enhanced Git so that users of the Remote System Explorer could access repositories created on remote servers in 2017. In 2014, the Eclipse Foundation reported that Git had become with most widely used source code management tool, with close to half of software developers reportedly using Git as their primary source code controls system.

Note: Access the full Git User Guide at : [http://wiki.eclipse.org/EGit/User\\_Guide](http://wiki.eclipse.org/EGit/User_Guide).

In your Eclipse Installation Details, you will find references to the Eclipse Git Team Provider. The Eclipse Git Team Provider provides access to the Git Perspective. Configurable elements of the Git Source Code Control System can be found in Window>Preferences>Team functions.

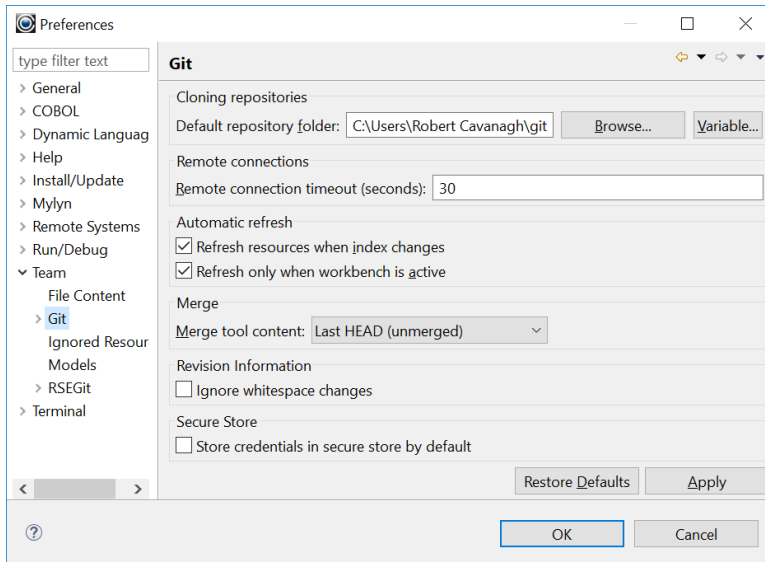
In the Developer Studio documentation, we will provide you with Getting Started scenarios for the 3 most common cases when using Git, which are:

- You wish to create a New repository, commit source to it, and maintain it as your source code repository.
- You are using Github, and wish to clone a repository for use with the Developer Studio.
- You already have a local Git repository, and you wish to integrate it with the Developer Studio.

## Configure Team Settings

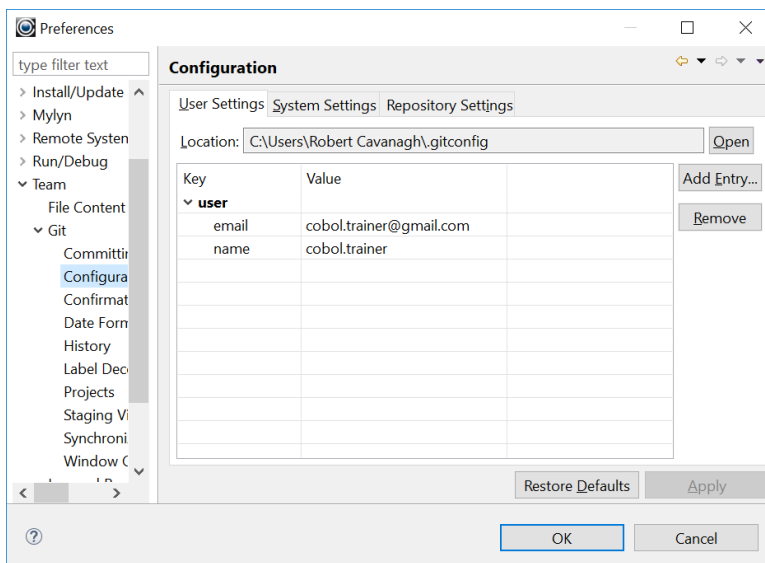
### Window>Preferences>Team>Git

For purposes of Cloning repositories (see “Clone a Git Repository”), set the default repository folder in Window>Preferences>Team>Git.



### Window>Preferences>Team>Git>Configuration

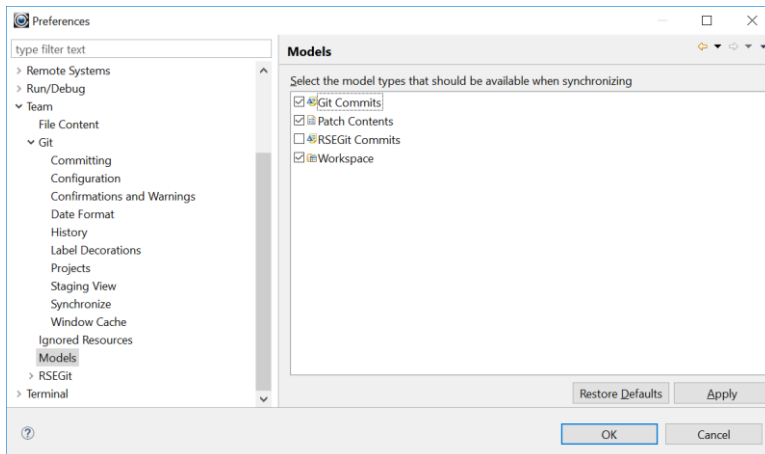
In the Configuration dialog screen, add entries for user.email and user.name.





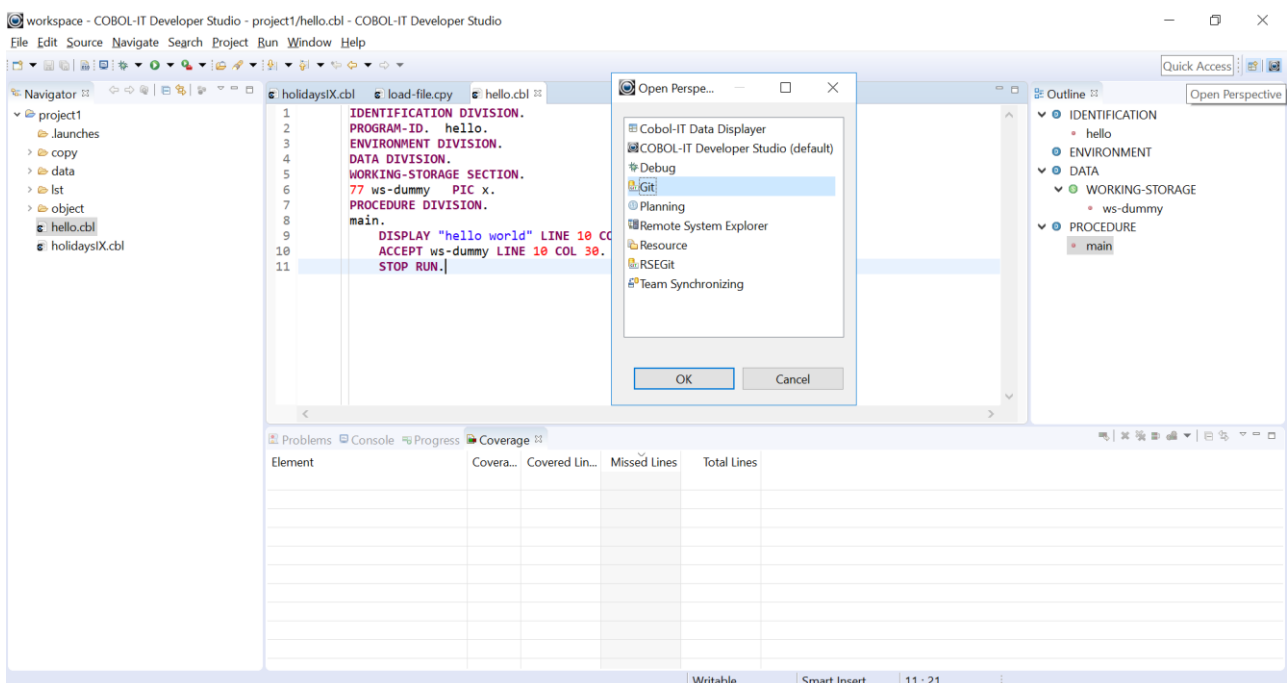
## Window>Preferences>Team>Models

On the Models screen, make sure that the “Git Commits” checkbox is selected and that the “RSEGit Commits” checkbox is de-selected.



## Open the Git Perspective

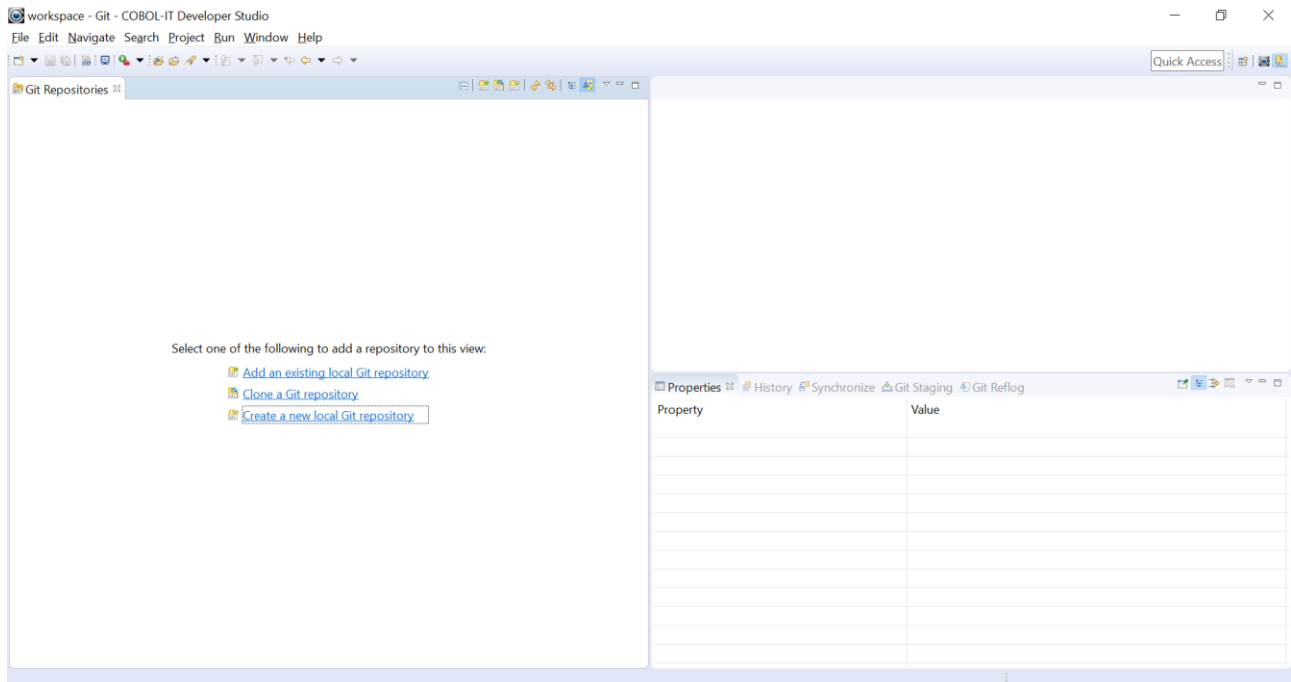
Click on the Open Perspective button on the Developer Studio toolbar, and then select Git from the Open Perspective window.





## The Git Perspective

The Git Perspective consist of the Git Repositories View, in the panel on the left, and the Properties, History, Synchronize, Git Staging and Git Reflog Views in the lower right.



When the Git Perspective is first opened, selections exist in the Git Repositories view for adding a repository. The selections are:

Add an existing local Git repository  
Clone a Git repository  
Create a new local Git repository

Allows use of an existing repository  
Clones an existing repository  
Creates a new repository

In this Getting Started document, we will exercise each of these, and look at some of the more commonly used Git functions. The full Git User Guide is accessible at :

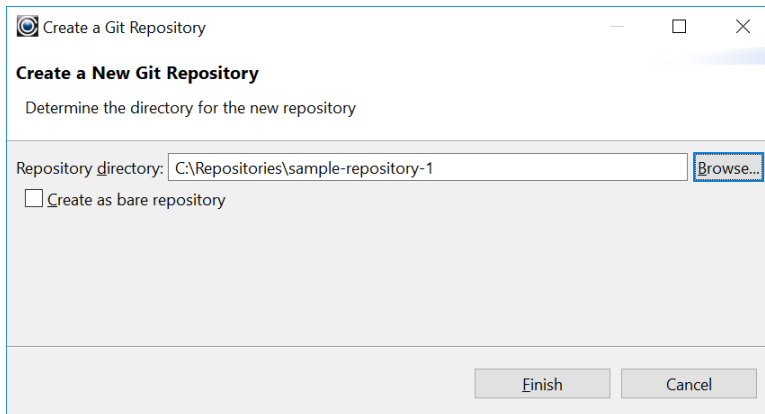
[http://wiki.eclipse.org/EGit/User\\_Guide](http://wiki.eclipse.org/EGit/User_Guide).



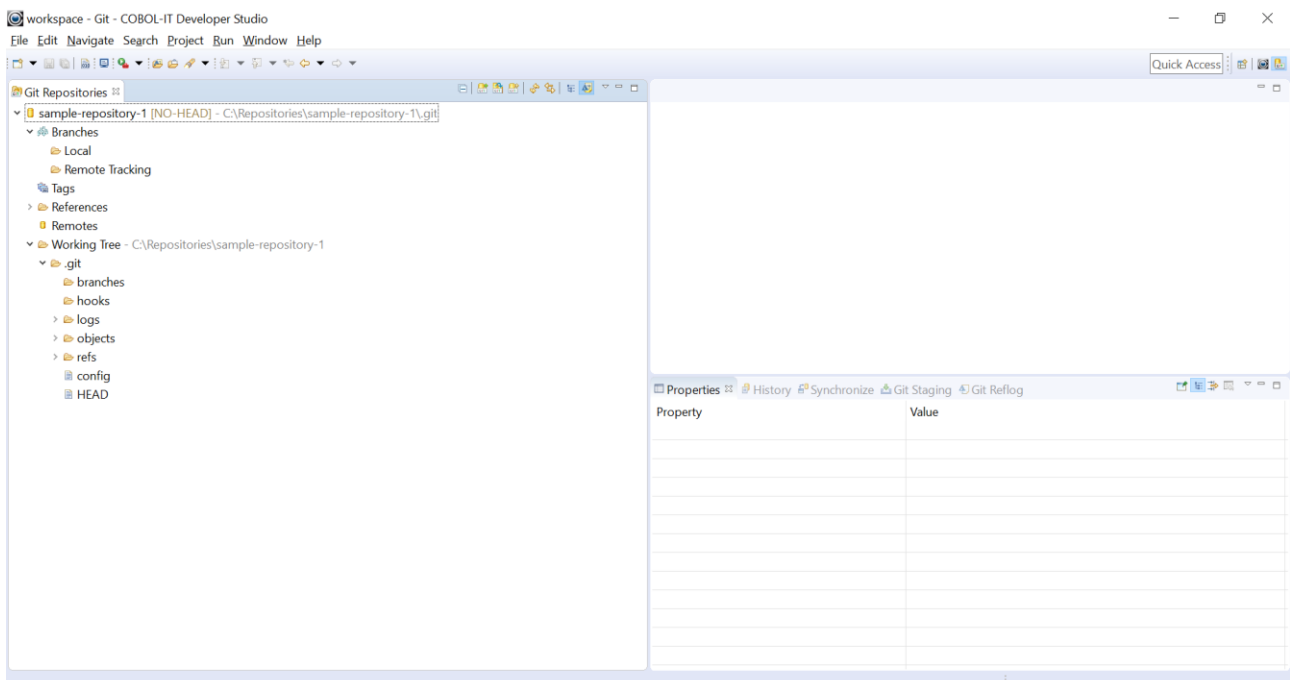


## Create a New Repository

In the Git Repositories Window, select "Create a new local Git repository". This will open a window titled "Create a Git Repository". First, you must set the Repository Directory. In the Repository directory entry-field, enter the full path to your repository directory. In our case, we will create a subdirectory in C:\COBOL folder called sample-repository. Then, we will click on the "Finish" button.



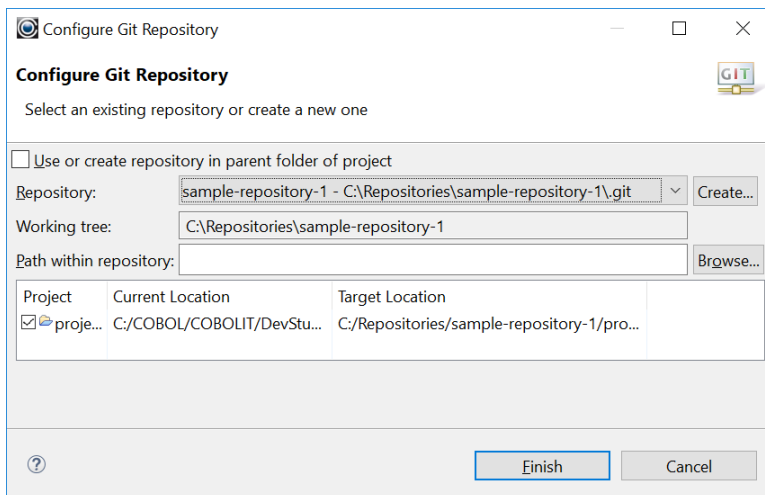
The new repository has been created.



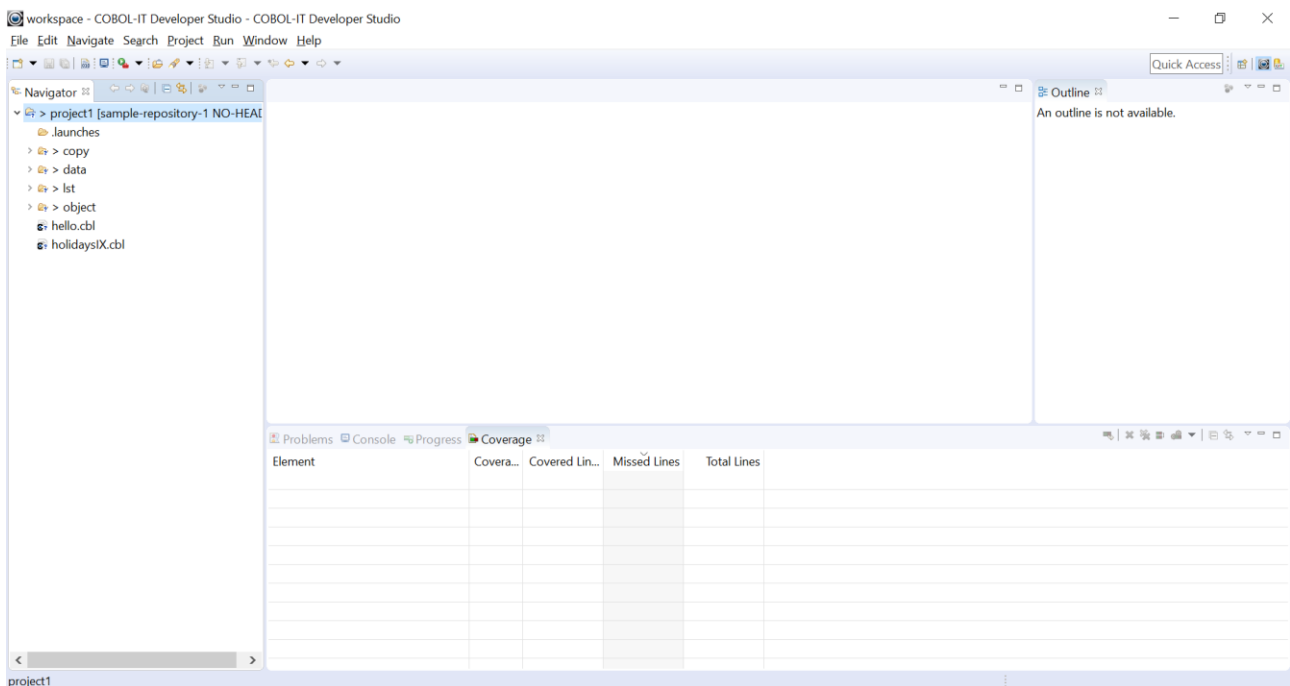
## Configure the new Git Repository

In the Developer Studio, select the project name in the Navigator window. Right-click, and select Team from the drop-down menu. From the subsequent drop-down menu, select "Share Project...". Click on the drop-down button on the Repository drop-down box and select the repository that was just created. Click on the "Finish" button to return to the Developer Studio.



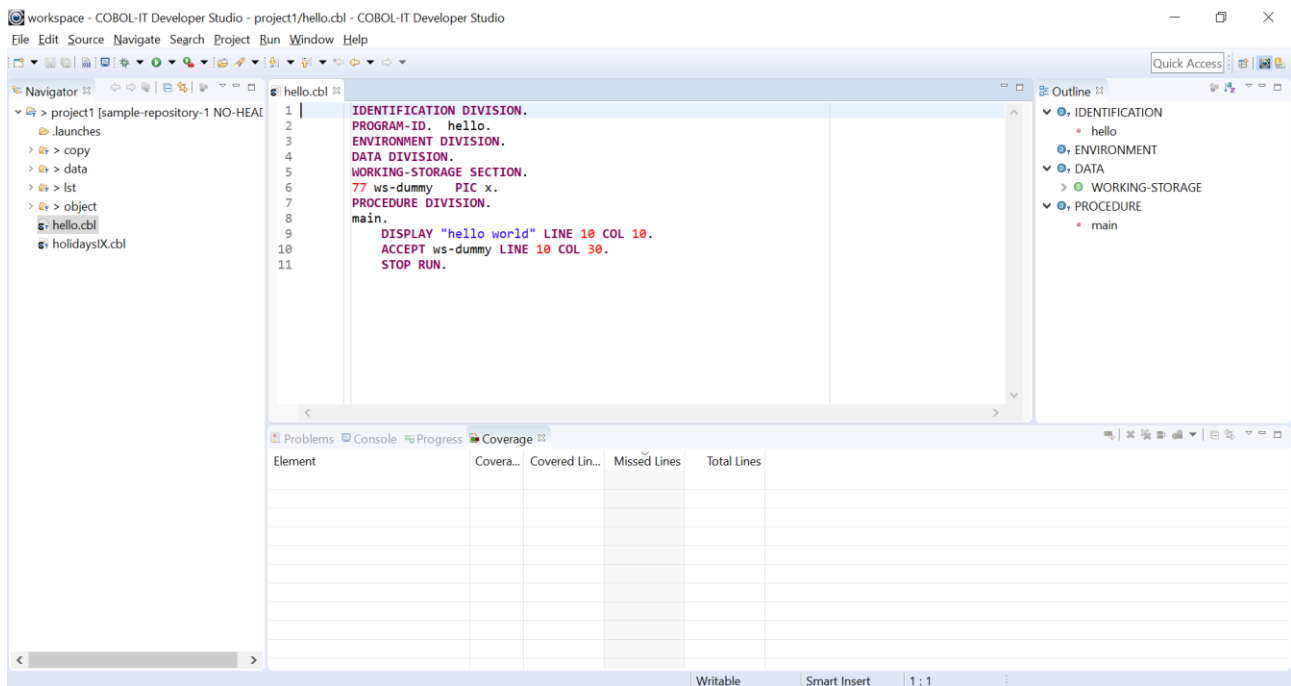


Your project has now been marked as being shared with our Git repository. At the project level, the label [git NO-HEAD] has been added. The Git icon has been added to the Project tree.



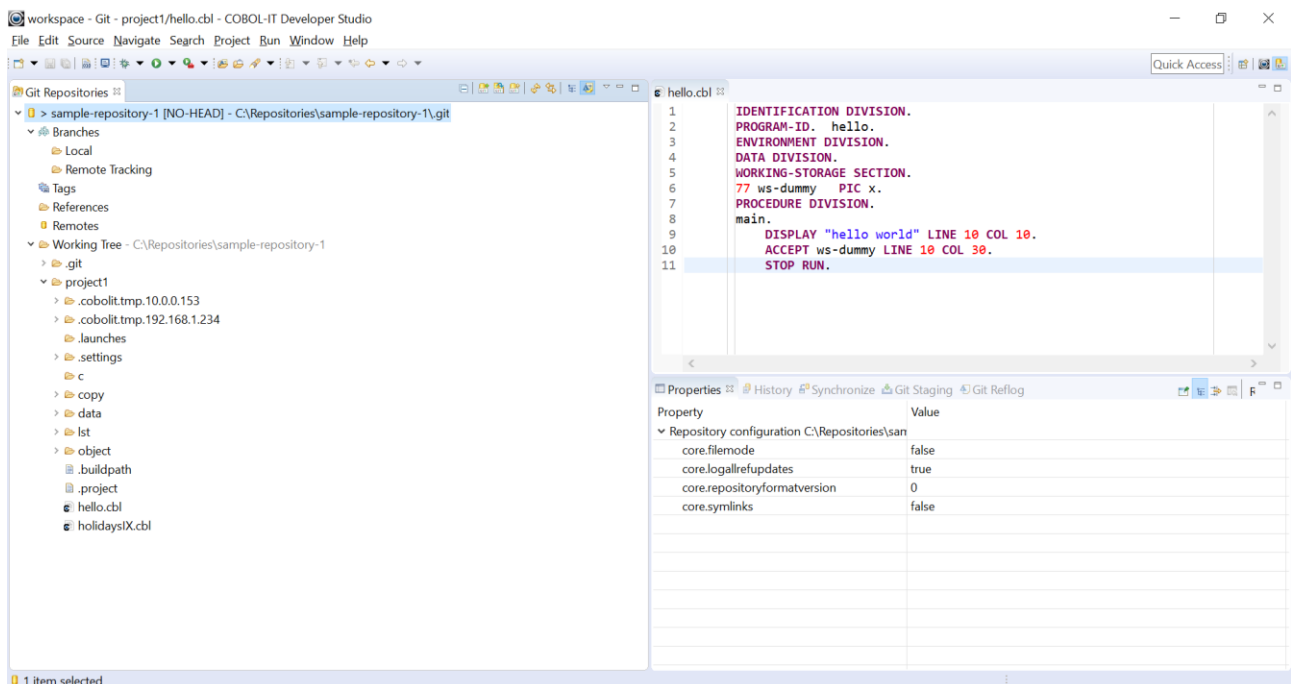
## Your Project in the Developer Studio Perspective

You will see label decorations on your source files and project in the Developer Studio Perspective. For details about the Git decorations, see [Window>Preferences>Team>Git>Label Decorations](#).



## Your Project in the Git Perspective

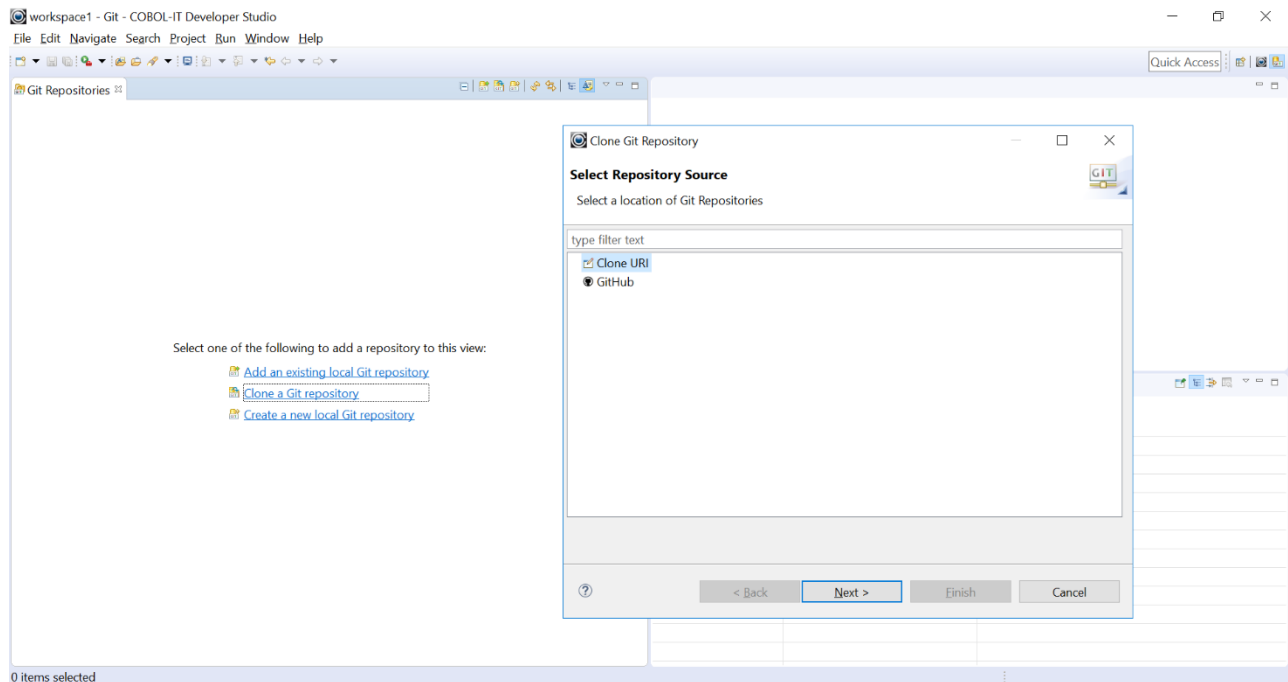
Project1 is now stored in the Git Perspective. It can be imported into an empty workspace. It can be shown in the Properties, History, Synchronize, Git Staging and Reflog views.





## Clone a Git Repository

In the Git Repositories Window, select "Clone a Git Repository". This will open a window titled "Clone Git Repository". Select the "Clone URI" option. Then, click on the "Next" button.



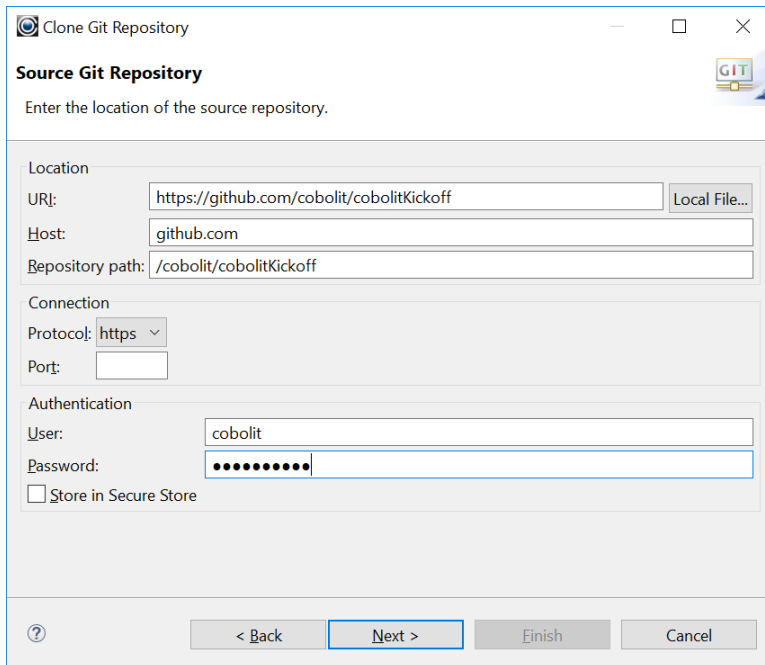
This will open the Source Git Repository Dialog.

## Source Git Repository

In the Source Git Repository Dialog window, enter the URI, Host, and Repository Path. In our example, we are hosting a Repository on github.com with a Repository path of /cobolit/cobolitKickoff.

In order to access this repository you have to go through an authentication which involves entering a User name and Password.





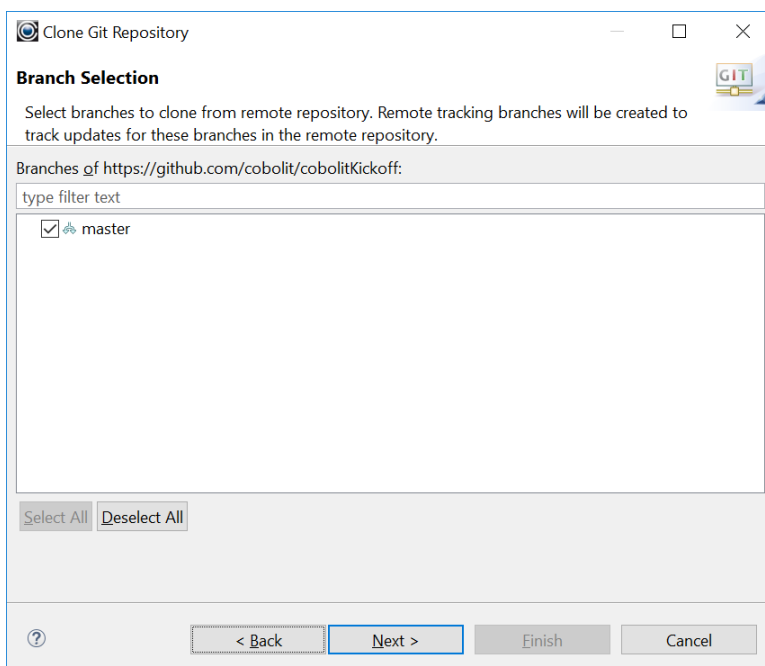
The 'Clone Git Repository' dialog box is shown. It has a title bar with a question mark icon, a maximize button, and a close button. The main title is 'Source Git Repository'. Below it, a subtitle says 'Enter the location of the source repository.' The dialog is divided into several sections: 'Location' with fields for 'URI:' (https://github.com/cobolit/cobolitKickoff), 'Host:' (github.com), and 'Repository path:' (/cobolit/cobolitKickoff); 'Connection' with a 'Protocol:' dropdown set to 'https' and a 'Port:' field; and 'Authentication' with a 'User:' field (cobolit) and a 'Password:' field (masked with dots). There is a checkbox for 'Store in Secure Store'. At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', 'Finish', and 'Cancel'.

From here, you will select a Branch to Clone, and select a location for the cloned repository. Then, we will use an Import wizard to import the project into the Developer Studio.

Click on the “Next” button to advance to the Branch Selection screen.

## Branch Selection

On the Branch Selection screen, select branches to clone from the remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.



The 'Branch Selection' dialog box is shown. It has a title bar with a question mark icon, a maximize button, and a close button. The main title is 'Branch Selection'. Below it, a subtitle says 'Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.' The dialog is divided into several sections: 'Branches of https://github.com/cobolit/cobolitKickoff:' with a 'type filter text' field; a list of branches with a checkbox and a branch icon next to 'master'; and buttons for 'Select All' and 'Deselect All'. At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', 'Finish', and 'Cancel'.



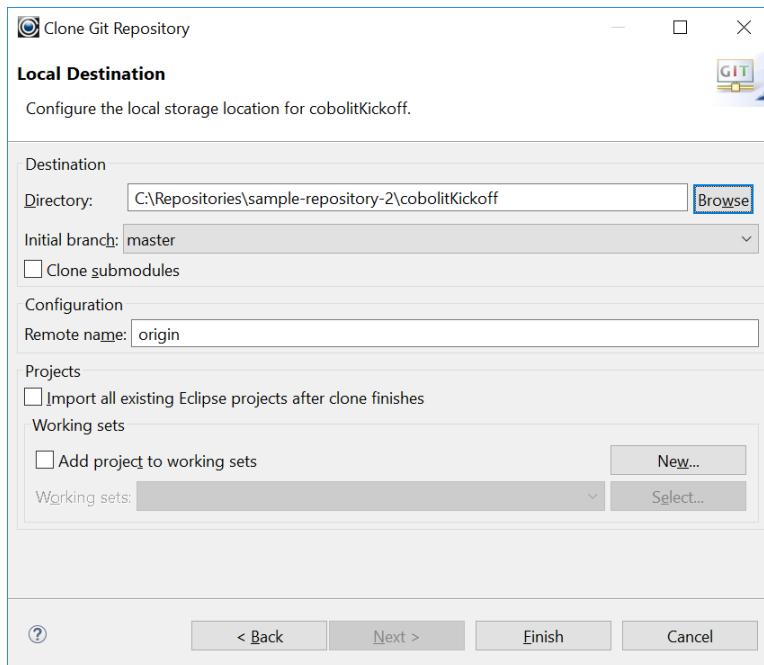


In our example we select the master branch.

Click on the “Finish” button to proceed to the Local Destination dialog window.

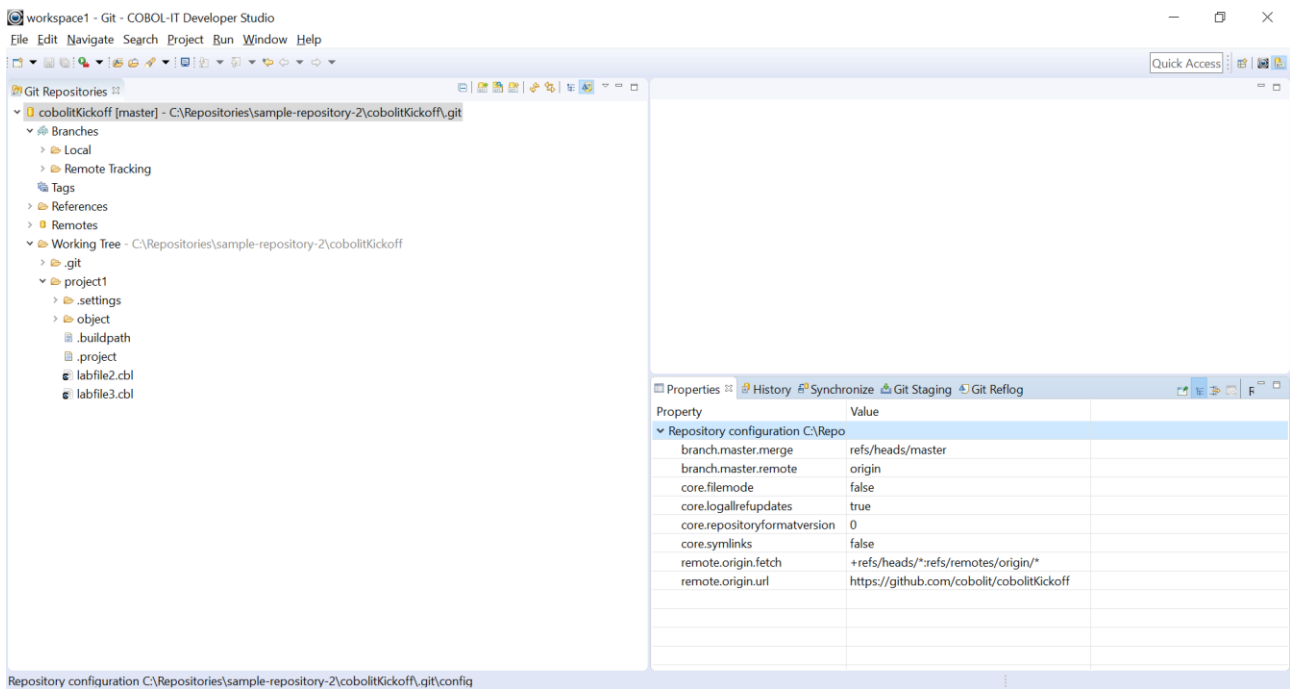
## Local Destination

In the Local Destination dialog window, enter the local directory where the repository will be cloned. The Local Destination dialog window contains the name of the Initial branch that has been previously selected. Click on the “Finish” button to update the Git Repositories Window.



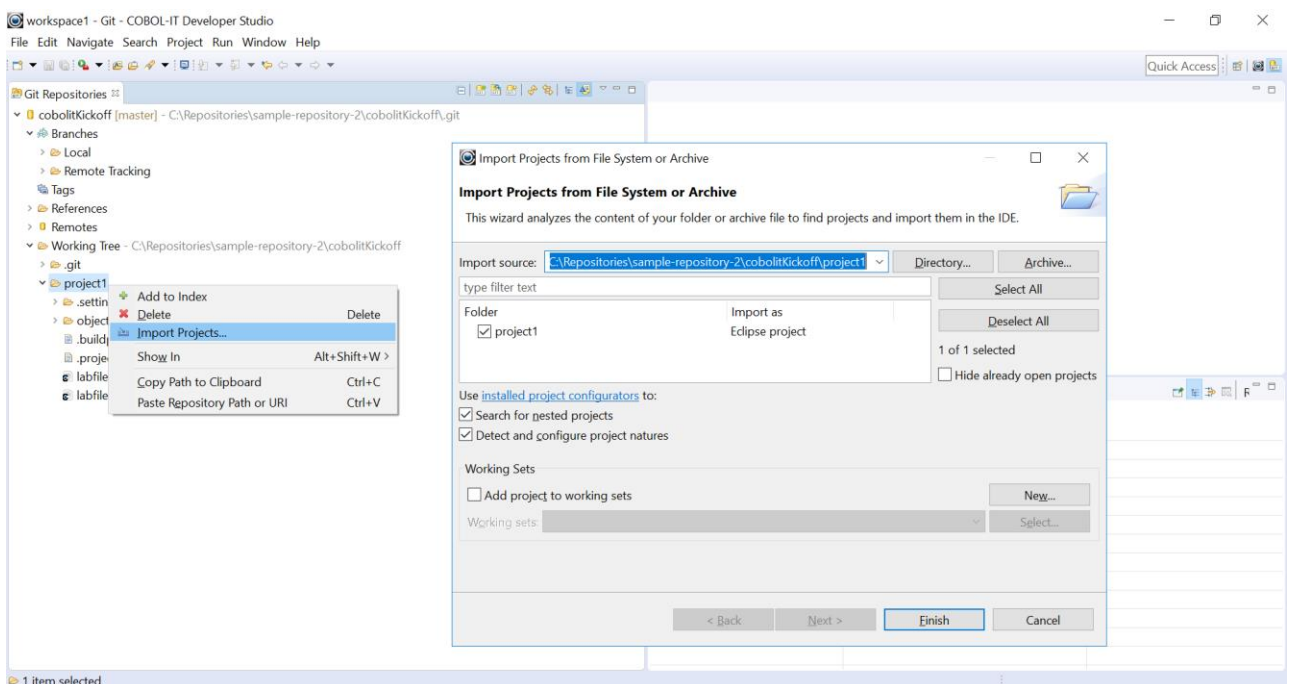
## The Cloned project in the Git Perspective

The project has been cloned in the default folder for cloned repositories



## Git Repositories>Import Project

In the Git Repositories Window, right-click on the cloned project and select “Import Projects...” to open the Import Project Wizard. First, you must select a wizard to use for importing projects. Select “Import existing Eclipse projects”. Then, select the project and click on the “Next” button to advance to the “Import Projects” window. You will see the project in the Import Projects window. Click on the “Finish” button.

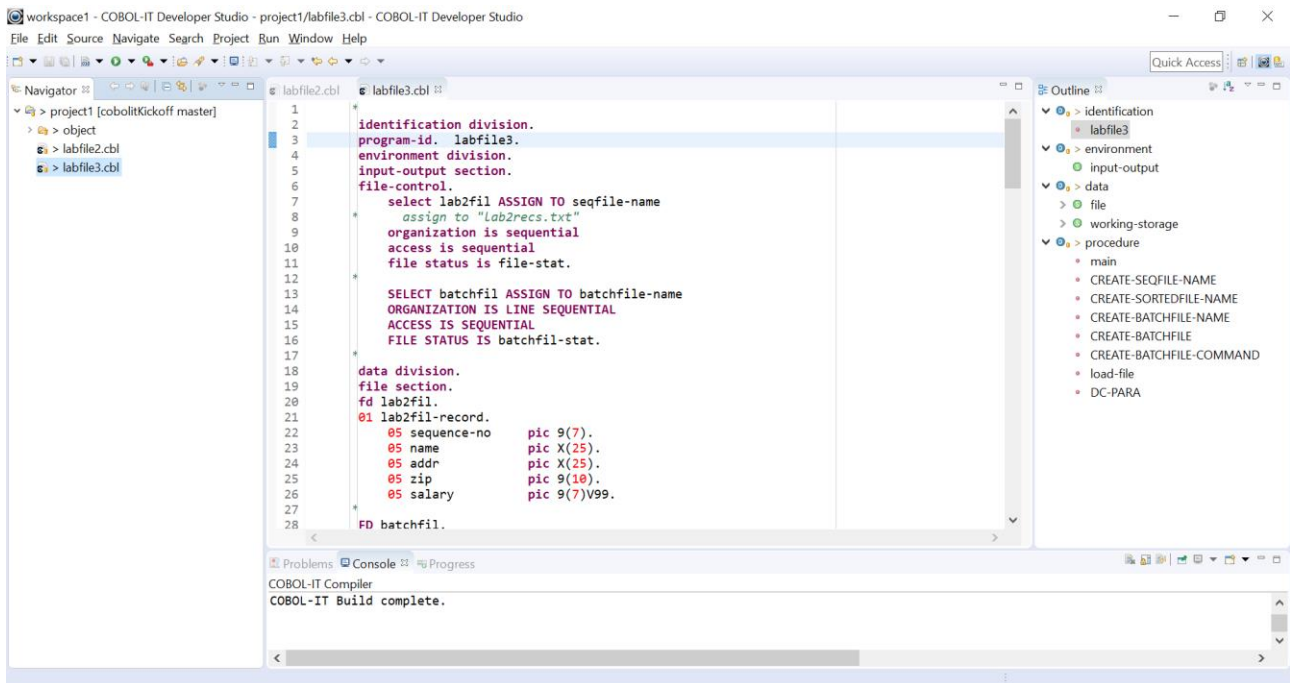




After clicking on the “Finish” button, open the Developer Studio Perspective, to access the Project and begin work.

## Access the Project in the Developer Studio Perspective

In the Developer Studio Perspective, you will see the Project Resources in the Navigator Window.



Now you can make a change to your source file, save the changes, and commit that change to the repository. To commit a source file to the Git Repository, right-click on the source file, and select Team from the dropdown menu. Then, click on the Commit function on the subsequent dropdown menu. This will open the Git Staging View, and you will see the source file you have changed listed as an “Unstaged Change”. Right-click on the source file name, and select “Add to Index”. The source file is now listed as a “Staged Change”. Enter a Commit message, and press the “Commit” button to commit the source code to the repository.



## Add an existing Git Repository

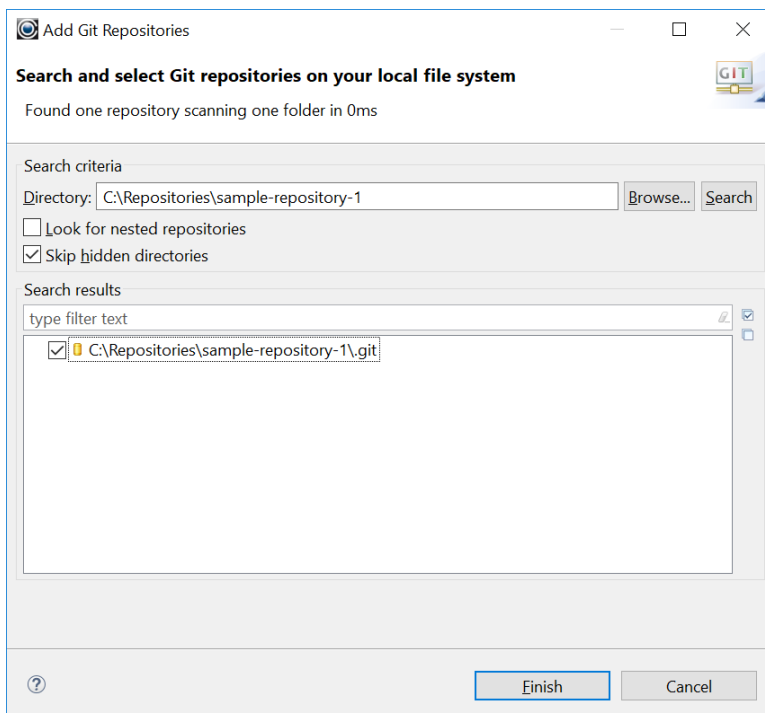
In the Git Repositories Window, select "Add an existing local Git Repository". This will open a window titled "Add Git Repositories".

First, you are prompted to Search and select Git repositories on your local file system.

First, enter a root directory that you wish to be searched for Git repositories.

Click on the “Search” button to populate the “Search Results” listbox.

Our search has located an existing Git Repository. Select the existing Git Repository by clicking in the checkbox to the left of the located repository. Then, click on the “Finish” button.

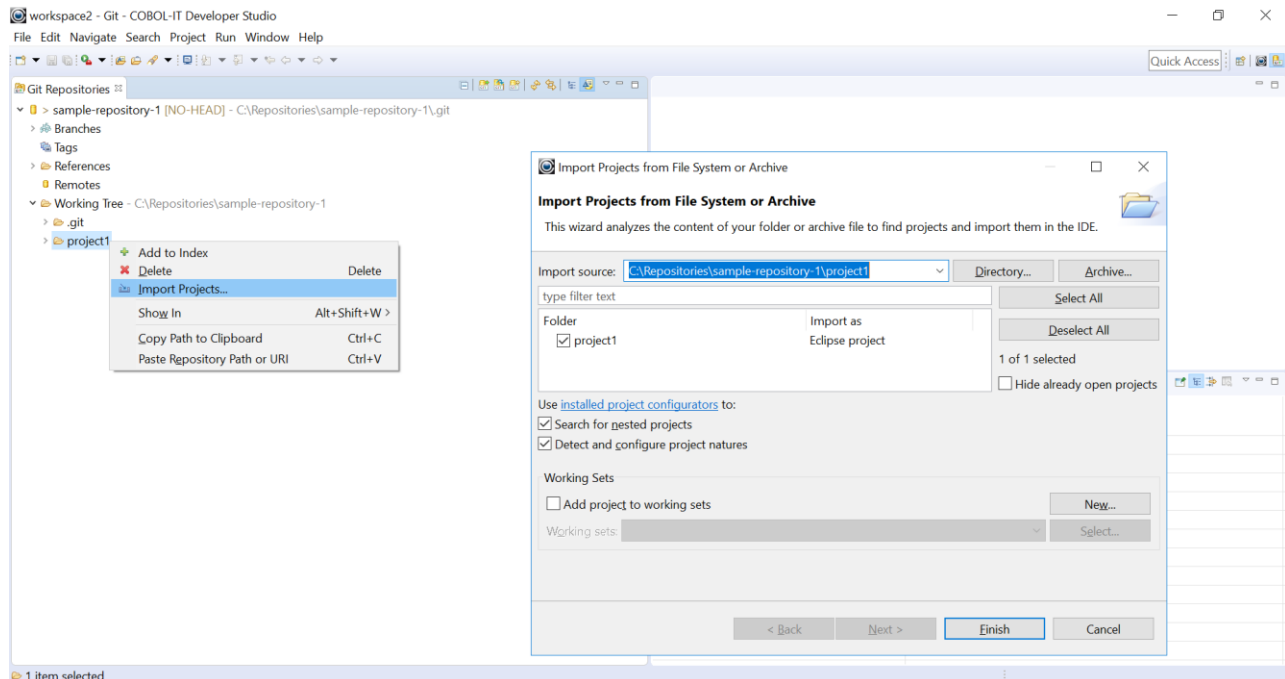


After clicking on the “Finish” button, you will return to the Git Repositories window, from which you can import the project into your Developer Studio.



## Git Repositories View>Import Project

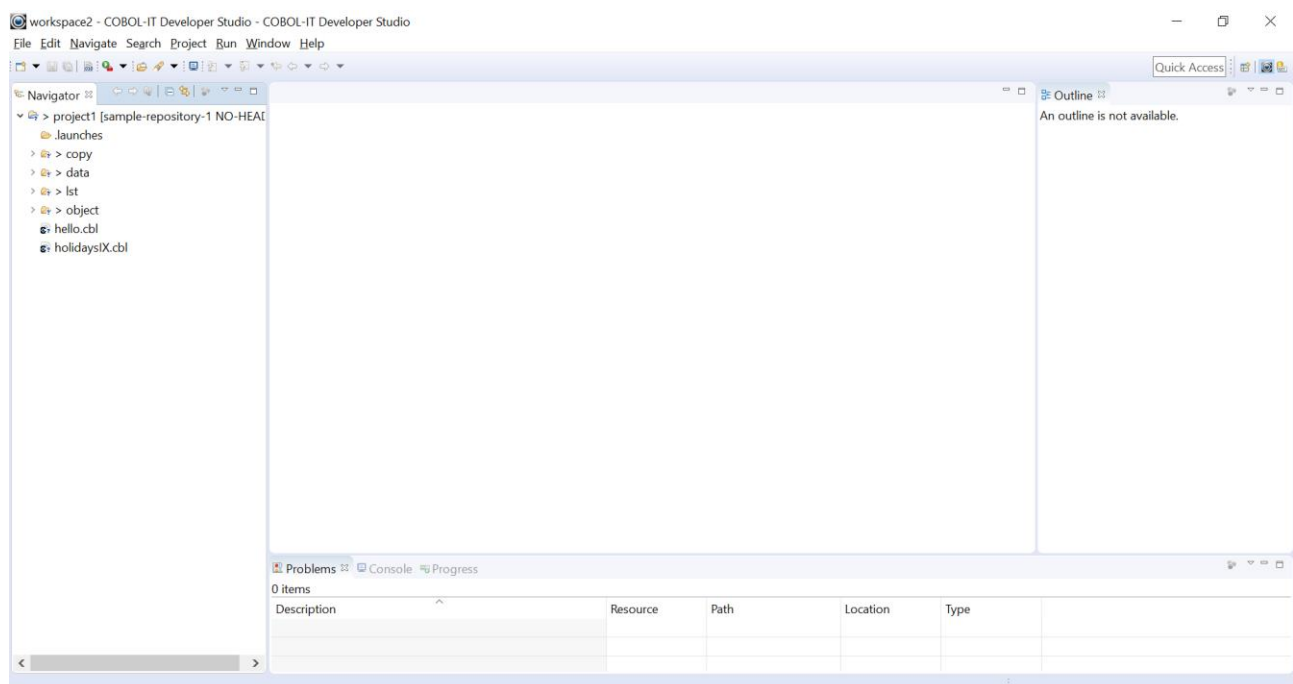
In the Git Repositories View, right-click the cloned project and select “Import Projects...”. From the dropdown menu select “Import Projects...” to proceed to the Import Wizard.



Click on the “Finish” button, open the Developer Studio Perspective, to access the Project.

## Access the Project in the Developer Studio Perspective

In the Developer Studio Perspective, you will see the Project Resources in the Navigator Window.



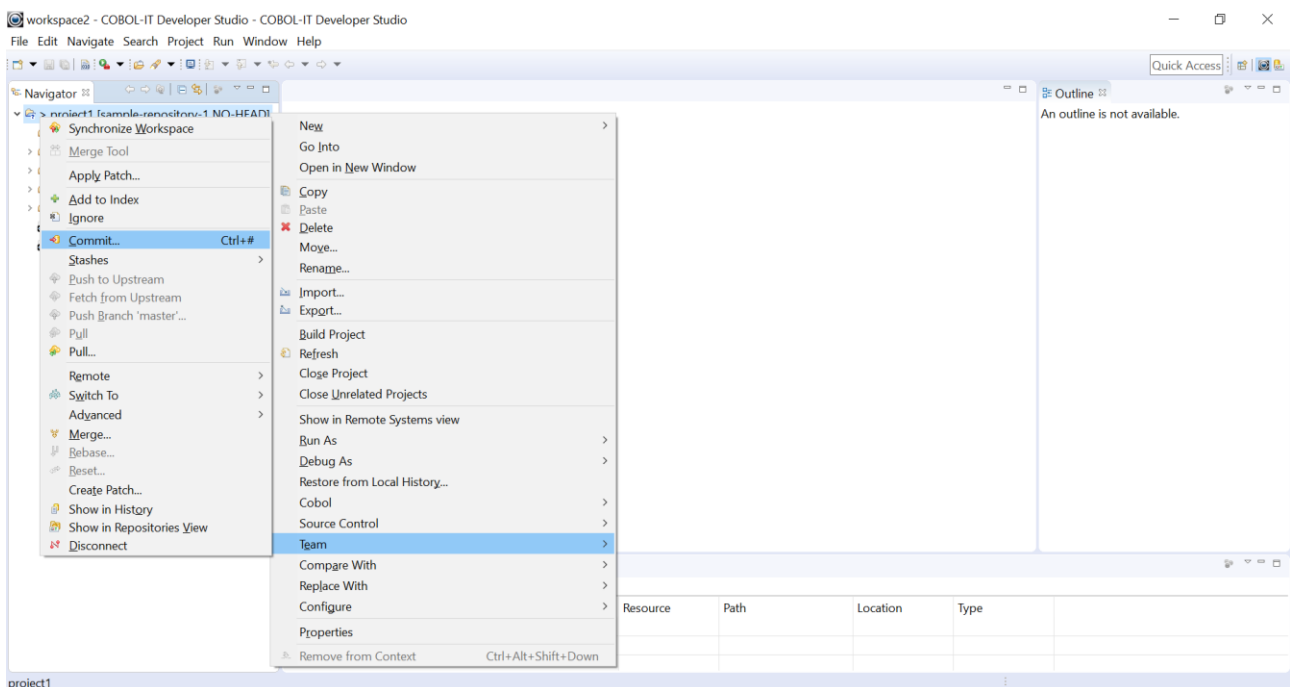


Now you can make a change to your source file, save the changes, and commit that change to the repository.

## Team>Commit

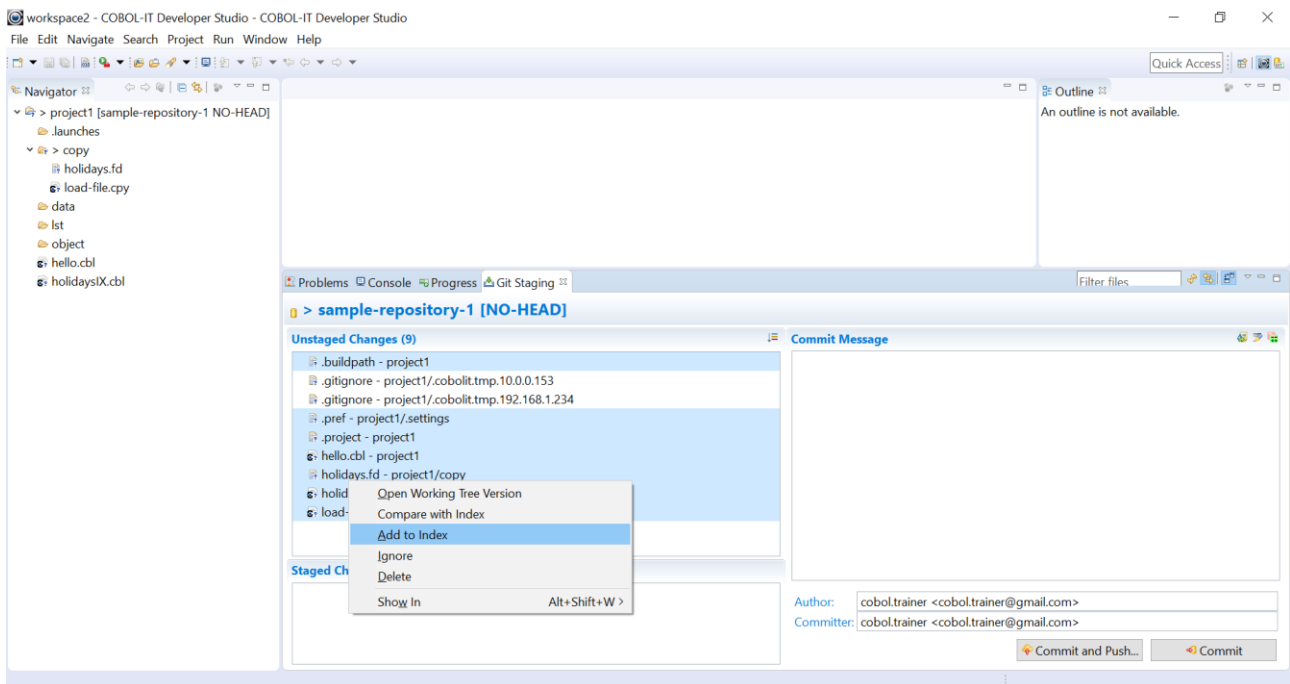
### Commit Code to the Repository

To commit your project files and source file to the Git Repository, right-click on the project, and select Team from the dropdown menu.

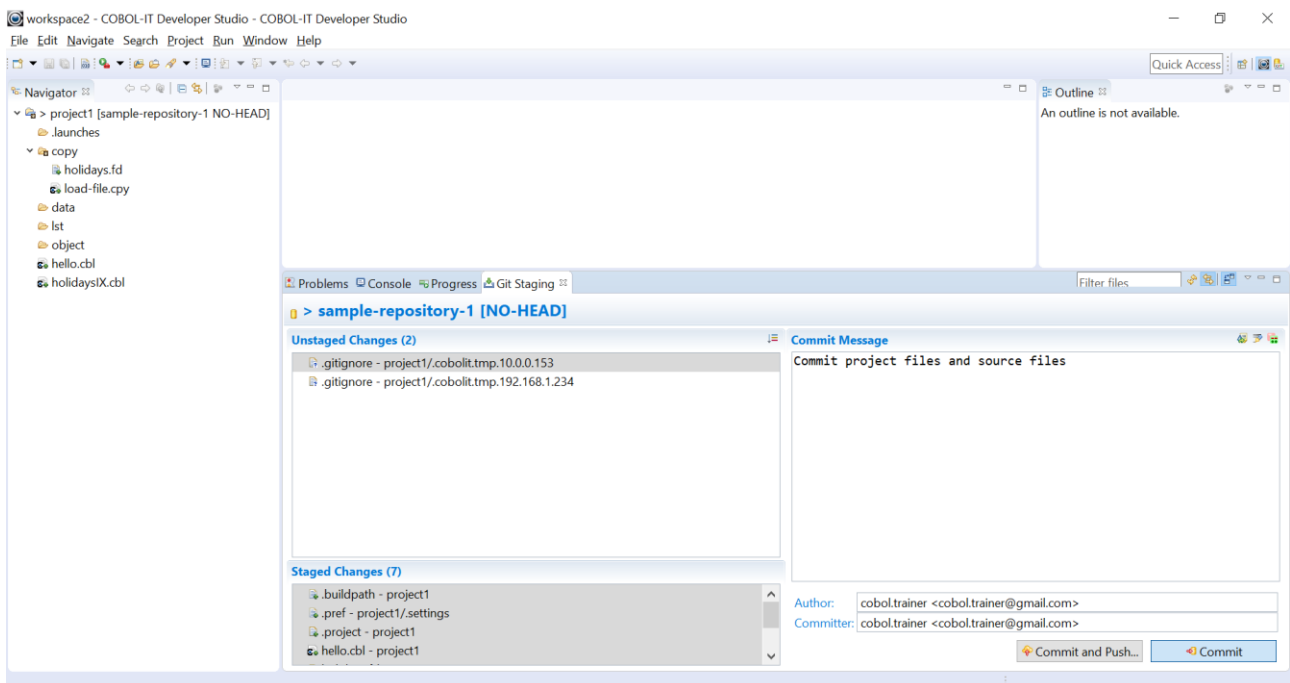


### Git Staging

This will open the Git Staging View. Your project files and your source file you have changed listed as an “Unstaged Changes”. Select the files, right-click, and select “Add to Index”.



The files are now listed as “Staged Changes”. Enter a Commit message, and press the “Commit” button to commit the source code to the repository. If you are working with a Cloned Repository, and wish to Commit changes back to the Origin, click on the “Commit and Push” button.

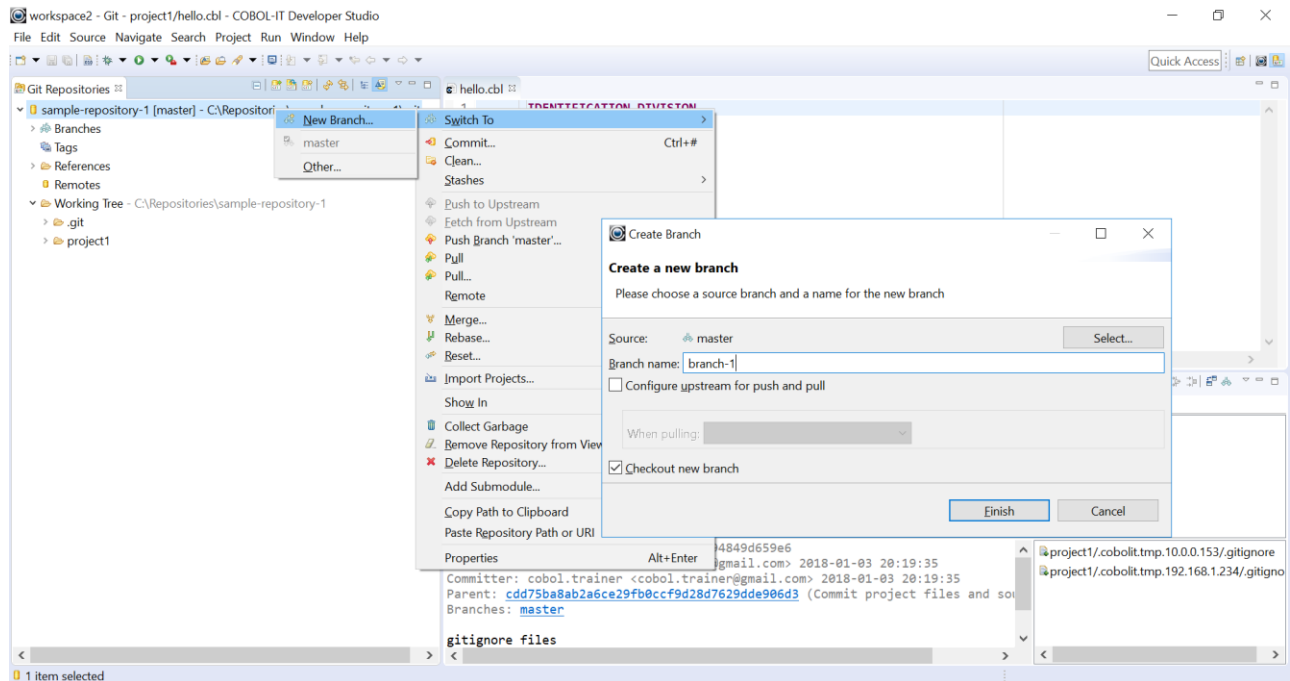




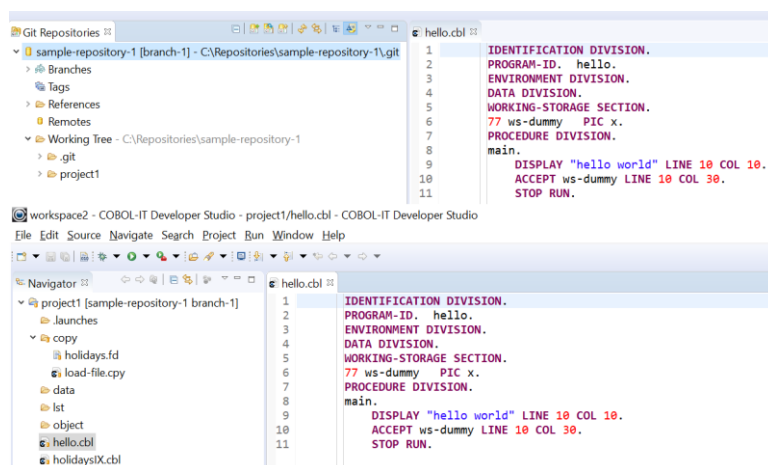
# Git Repositories View

## Switch to>New Branch

To Create a New Branch, or Switch to a Branch, select the Repository in the Git Repositories view, right-click, and select “Switch To” from the dropdown menu and “New Branch” from the subsequent dropdown menu.

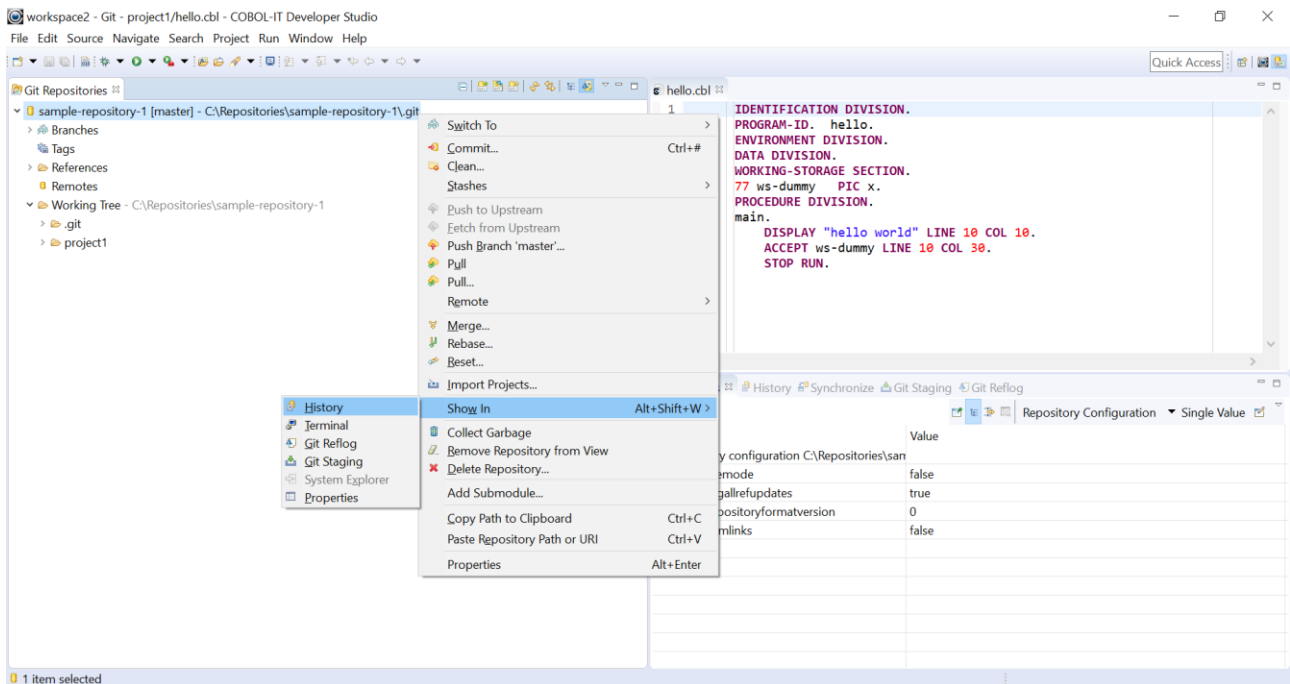


The interface creates branch-1 and performs a checkout on branch-1. The Git decorations in the Git Perspective and in the Developer Studio Perspective include the branch-1 label:

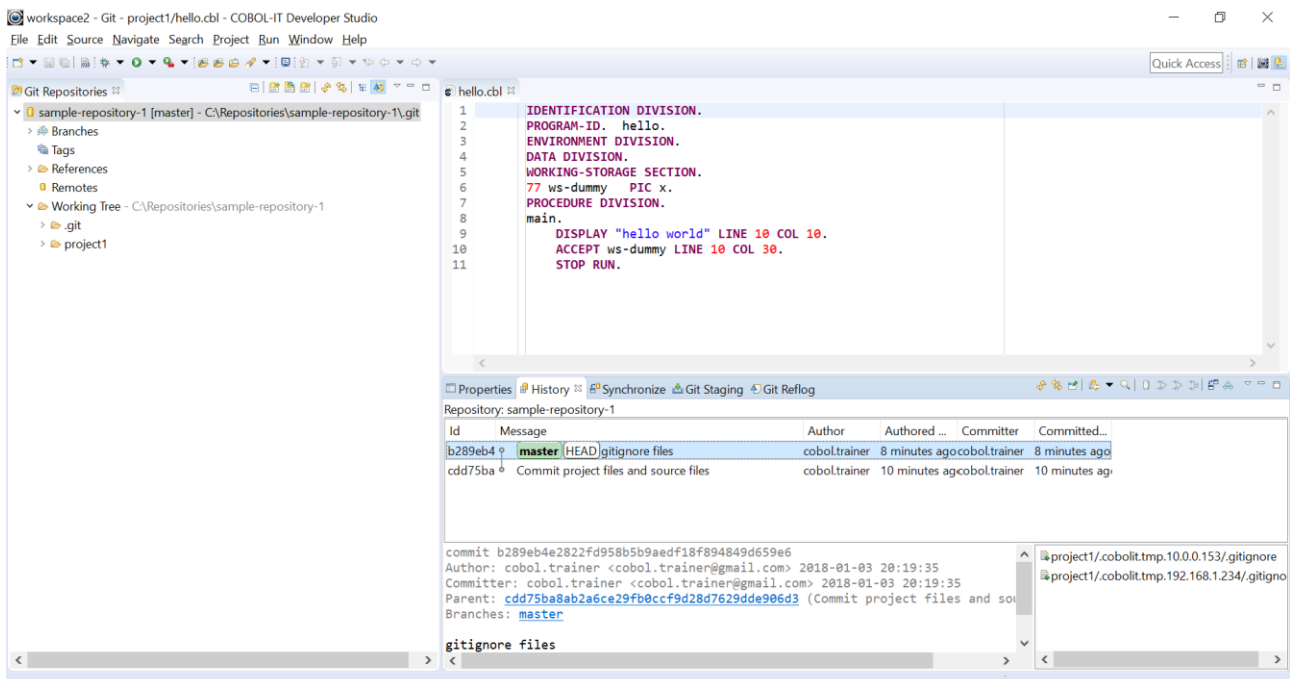


## Show in > History View

To view the Repository in the History View, select the repository by clicking on it, then right-click and select “Show In” from the dropdown menu, and “History” from the subsequent dropdown menu.



The Repository is shown in the History View:





## Using RSE Git for Source Code Control

RSE Git functionality requires an RSE connection to a Server on which COBOL source code is hosted in an existing Git repository. Inside the Remote System Explorer (RSE) connection, users can create a remote project using this existing COBOL source, and have access to Git functionality.

### Setup for use of RSEGit

#### Remote Server>COBOL Source Folder

For our sample we have a COBOL source folder called holidayprj, in which we have folders called data and object, and we have the following source files:

```
holidaysIX.cbl  
holidaysREL.cbl  
holidaysSEQ.cbl  
testit.cbl
```

#### Remote Server>COBOL Source Folder>git init

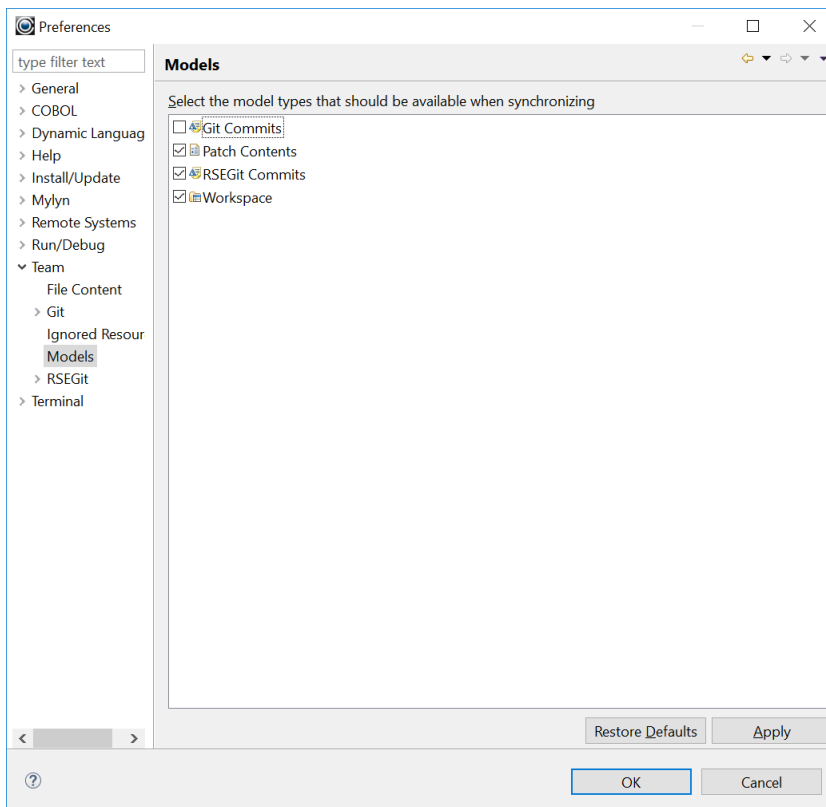
First, we will enter the COBOL source folder, and create and configure a git repository. We can do this by running the commands:

```
>git init  
>git config user.email "cobol.trainer@gmail.com"  
>git config user.name "COBOL Trainer"
```

#### Developer Studio>Window>Preferences>Team>Models

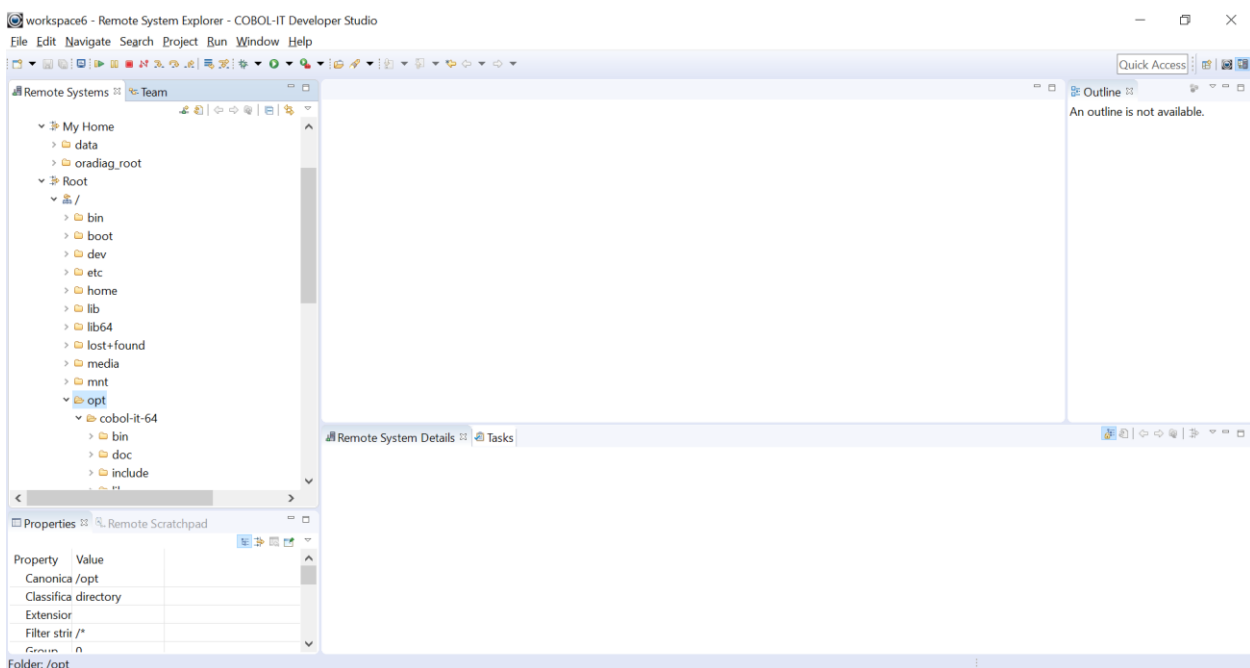
To configure the Developer Studio for use with RSEGit, navigate to the Window>Preferences>Team>Models dialog screen. On the Models dialog screen, make sure that the "Git Commits" checkbox is de-selected and that the "RSEGit Commits" checkbox is selected.





## Remote System Explorer>Establish a remote connection

RSEGit provides users of the Remote System Explorer (RSE) capabilities of the Developer Studio with the ability to use Git for Source Code Control. To begin using RSEGit, first establish a remote connection through the Remote System Explorer.





### Developer Studio>Create a project at existing location

In the Developer Studio Perspective, select File>New>Project. In the “New Project” dialog, expand the “COBOL” entry, and select “COBOL Project”. Click on the “Next” button proceed to the “Create a COBOL Project” dialog.

#### File>New>Project

Select File>New>Project, and then select COBOL>COBOL Project from the New Project dialog box. This will open the “Create Project” dialog box.

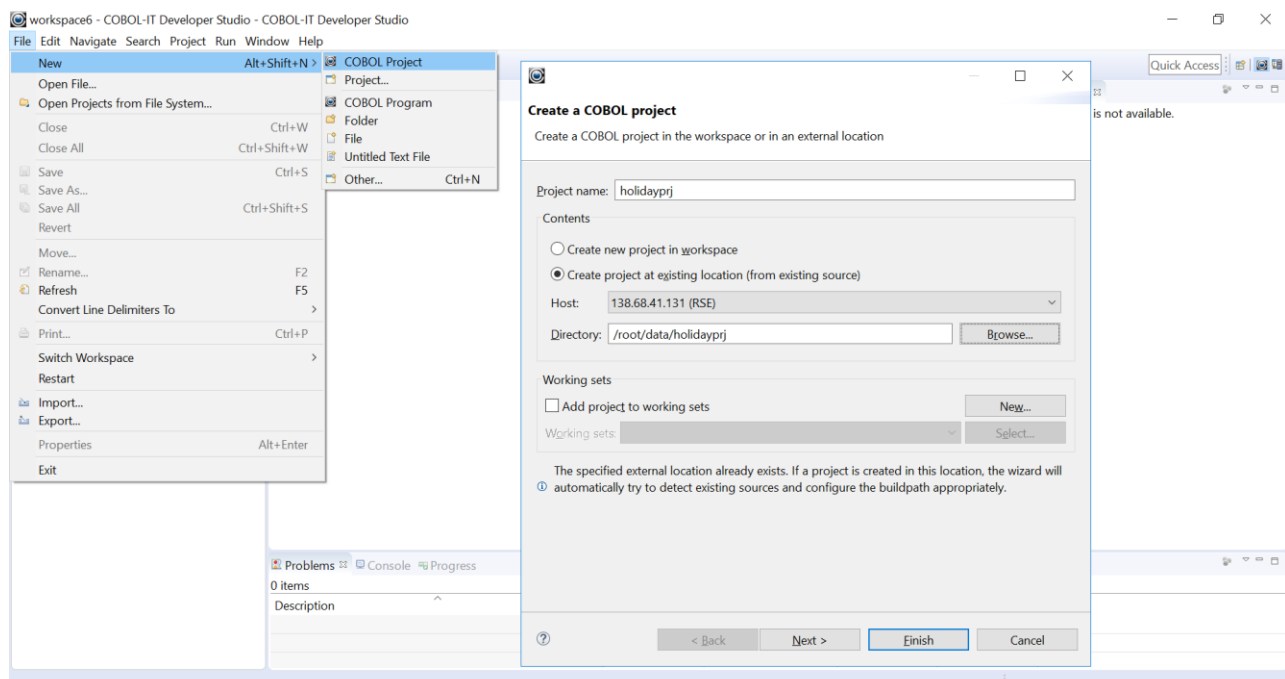
#### Create a COBOL Project

In the “Create a COBOL Project” dialog screen, name the project “holidayprj”. Select the “Create project at existing location” radio button.

Drop down the “Host” dropdown box and select the Remote host notation. Then, browse to the directory in which the COBOL source files are located and in which the Git repository has been created.

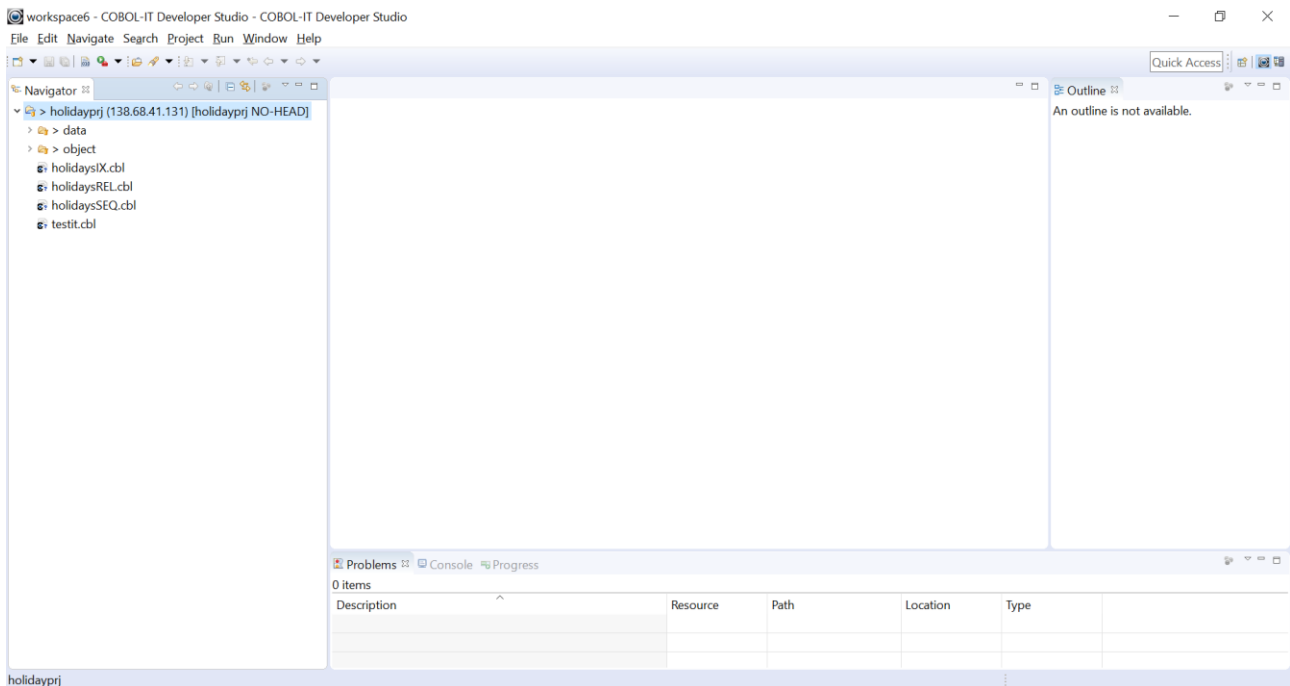
Click on the Finish button to create the new project.

The Project is created, and Git labels are applied to the project files.



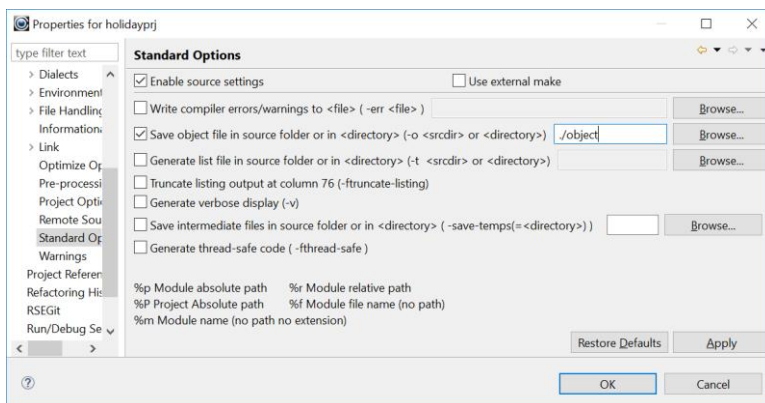


The Project is created and Git label decorations are applied to the project files.



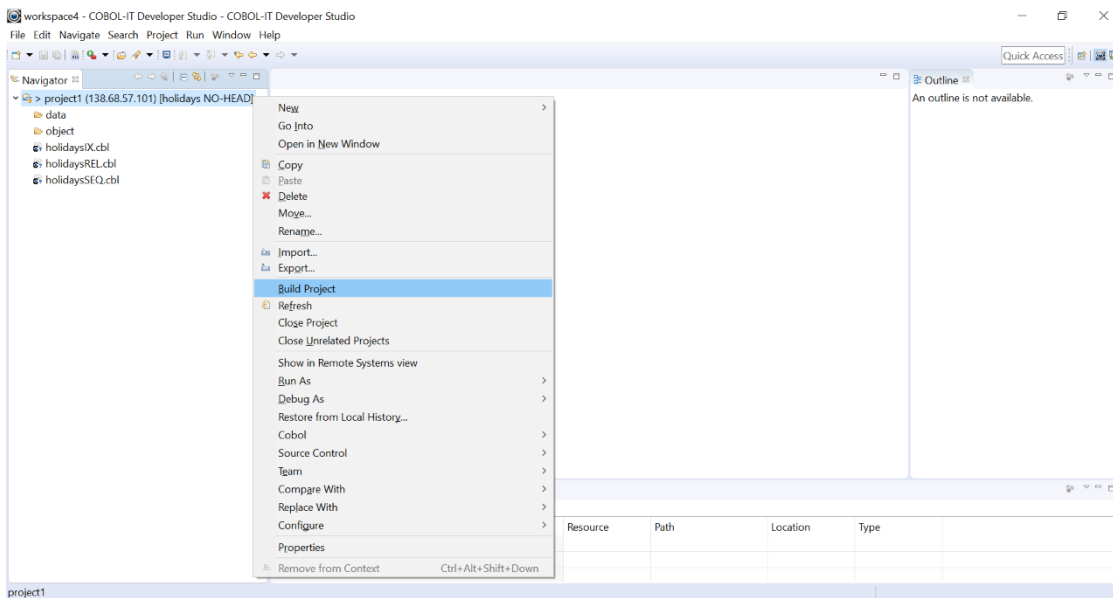
### ***Project>Properties>COBOL Properties>Standard Options***

Select the -o compiler flag option, and enter ./object into the corresponding entry-field.



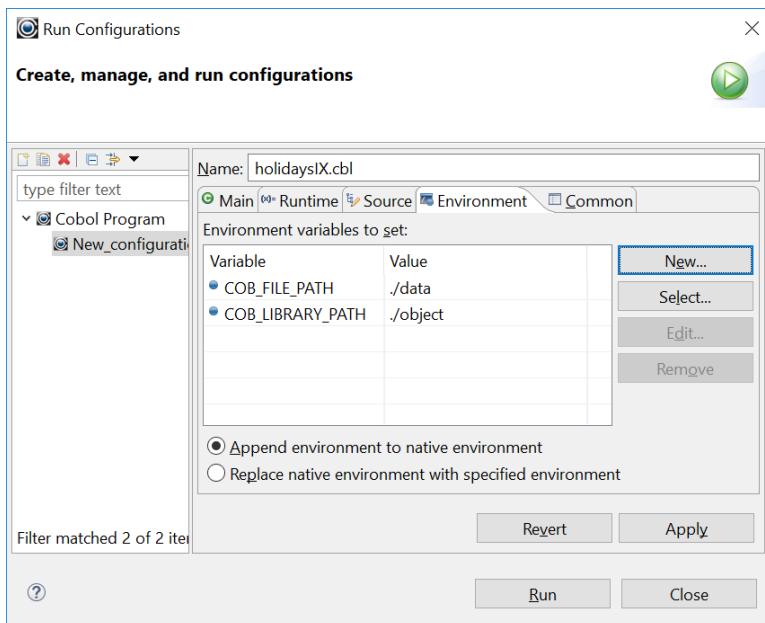
### ***Build the Project***

Select the project, and right-click to open a drop-down menu. From the drop-down menu, select “Build Project” to build the project.

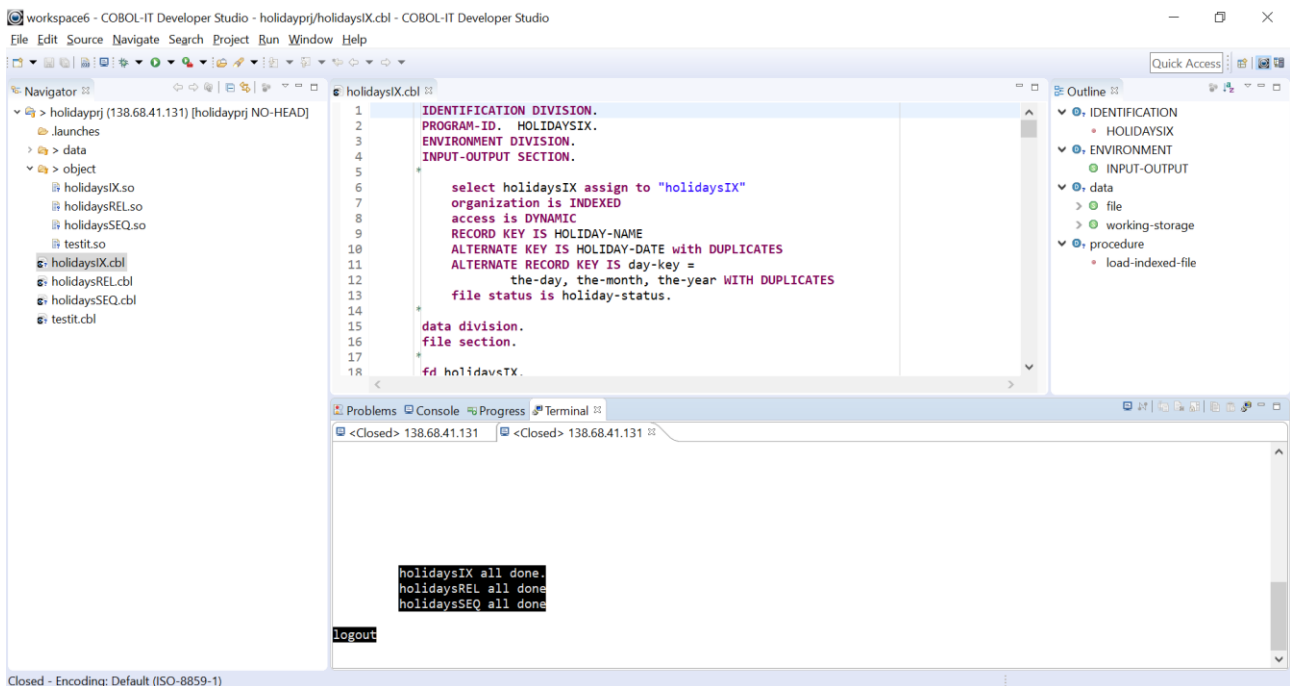


## Run the Project

Select `holidaysIX.cbl`, right-click and select `Run As...` from the drop-down menu to open the Run Configuration wizard. We have named our configuration `holidaysIX.cbl`. On the Main tab, verify that `holidaysIX.cbl` in `project1` is selected. On the Runtime tab, verify the runtime settings. On the environment tab, set `COB_FILE_PATH` to `./data` and `COB_LIBRARY_PATH` to `./object`. Click `Apply`, and then click on the `Run` button.

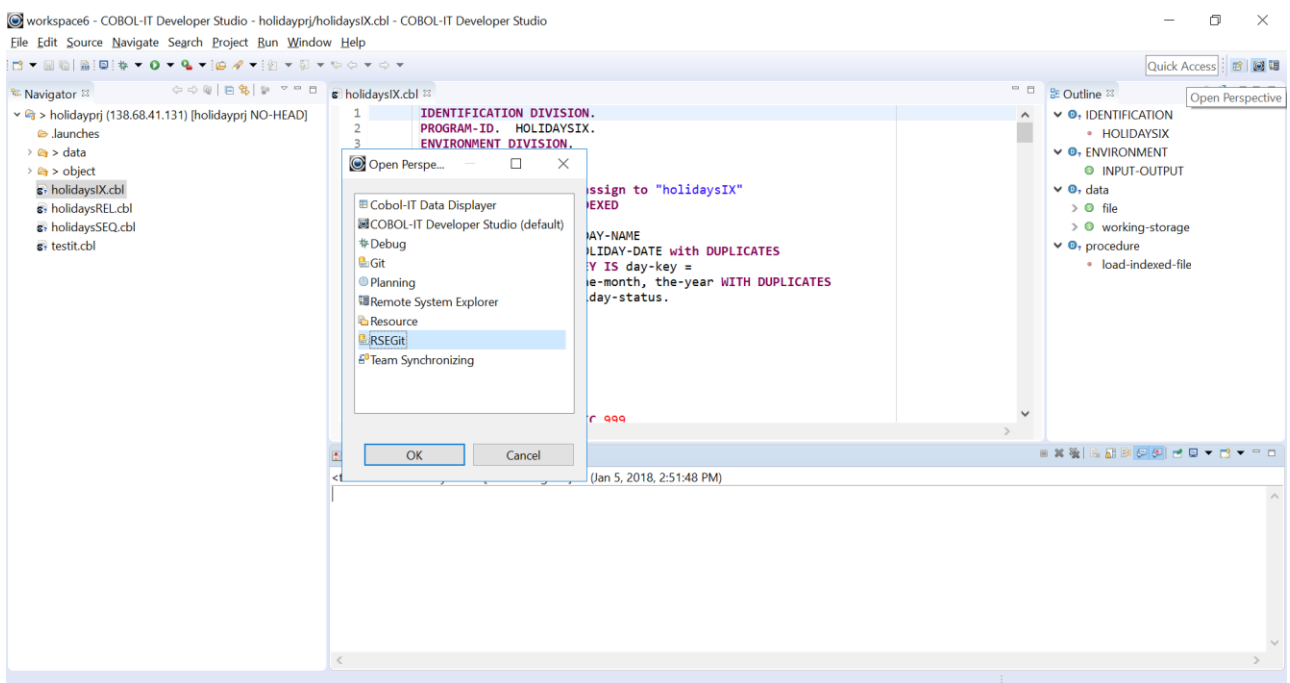


The program will run, creating indexed, relative and sequential files. We see our output in the Terminal window.



## Developer Studio>Open the RSEGit Perspective

Open the “Open Perspective” dialog window and select “RSEGit”. Click OK.



## RSEGit>RSEGit Repositories View

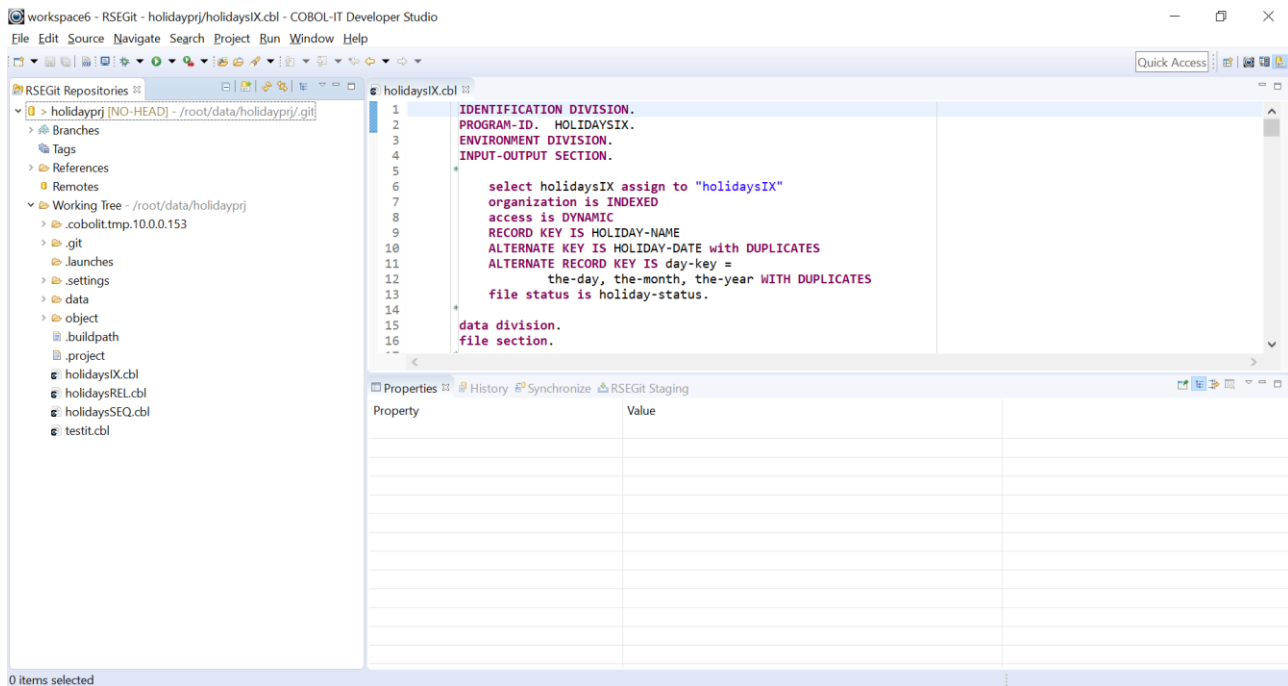
The RSEGit Perspective opens with the RSEGit Repositories view in the Panel on the left, and tabbed views for History, Synchronize and RSEGit Staging in the lower right. When the RSEGit Repositories view is empty, there is a functional link titled “Add an existing Git repository”, which can be used to locate and open an existing Git repository with Browse functionality.







In our case, we created a COBOL project in a source location that included a git repository, so the COBOL source files have been listed in the Git Repositories window.

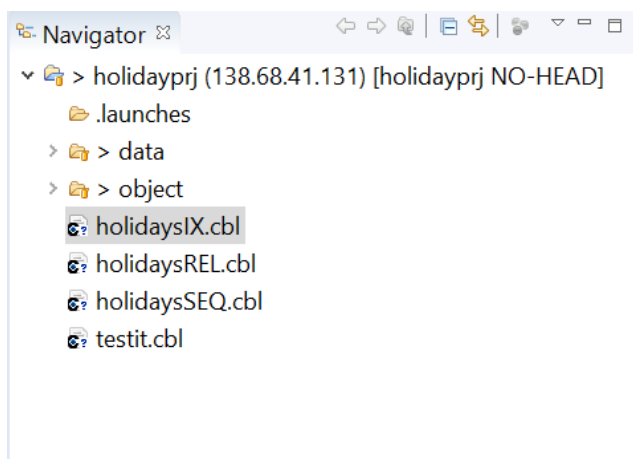


Setup is complete now, and you are ready to work in the Developer Studio using RSEGit.

## Committing Files

### The Navigator View of the COBOL Project>Untracked text

Switch to the Developer Studio perspective. The new project is displayed in the Navigator Window, and the project is decorated with Git label decorations. The files are marked with the “Untracked text” decoration. For details about Git Label Decorations, see [Window>Preferences>Team>Git>Label Decorations](#).

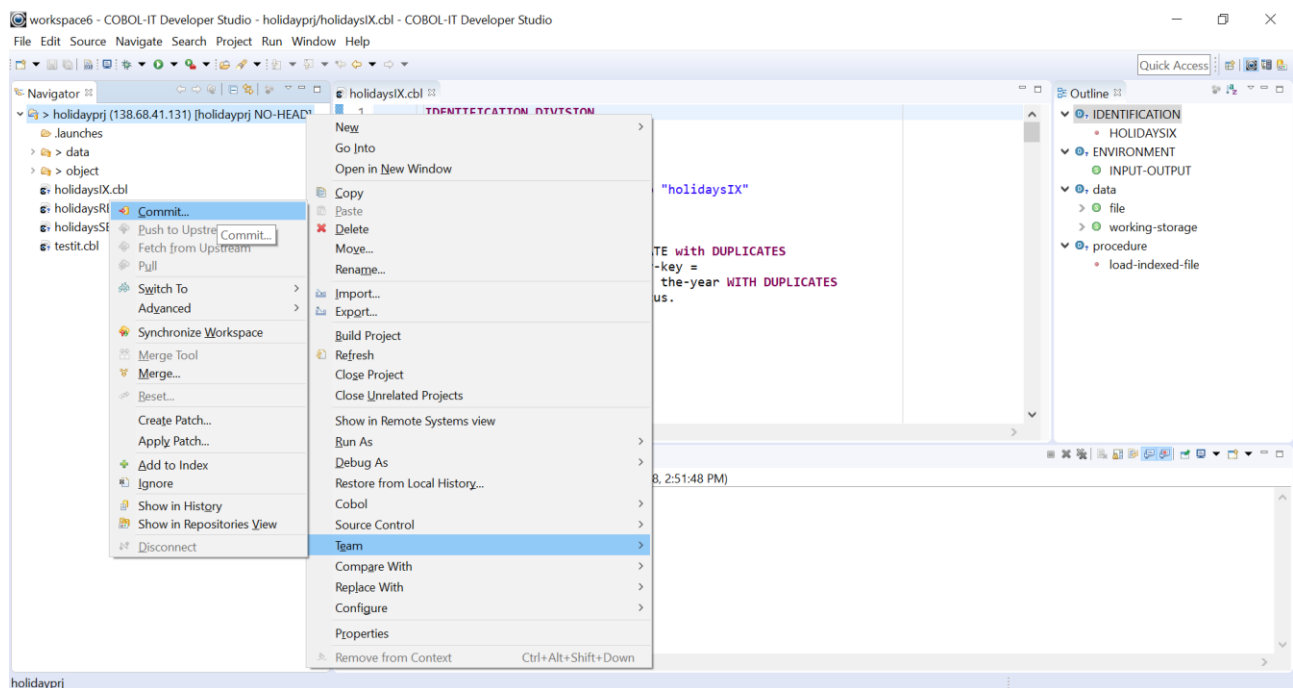




## [Project]>Team>Commit>RSEGit Staging

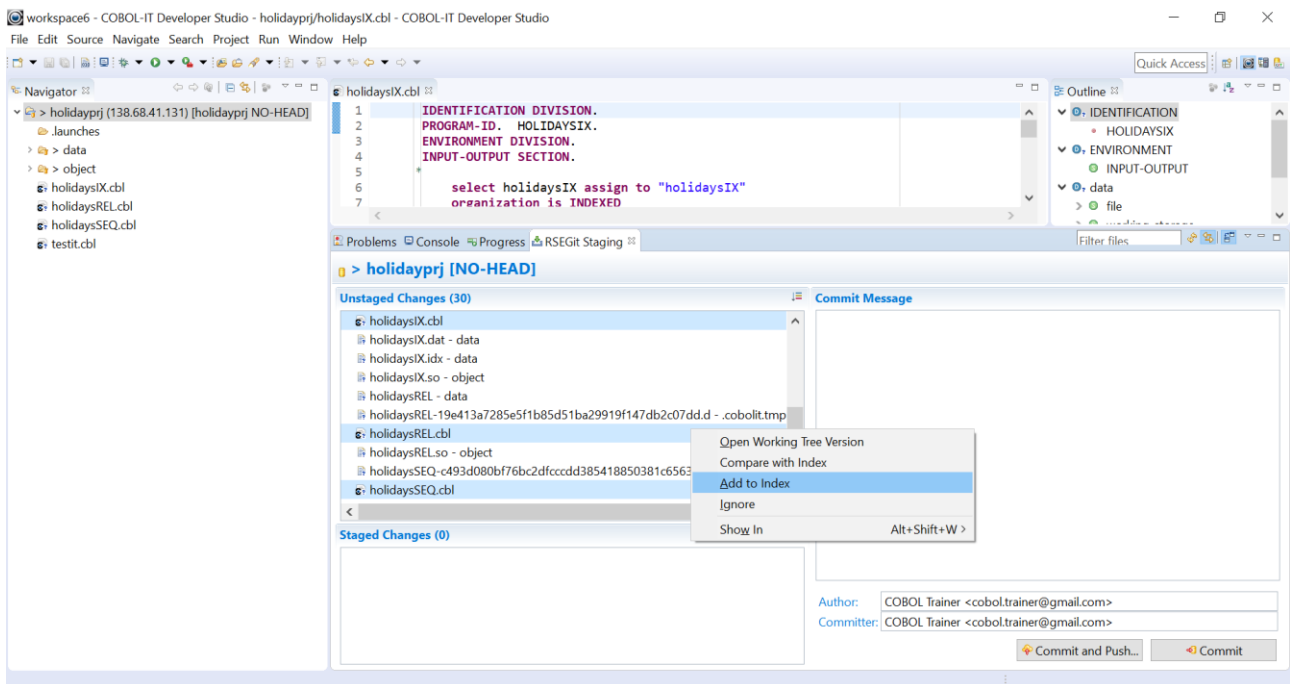
Right-click on the project listing, select Team from the dropdown menu, and Commit... from the subsequent dropdown menu to commit your project files to the repository.

The Team>Commit function at the project level causes the project files to be displayed in the RSEGit Staging view as “Unstaged Changes”.



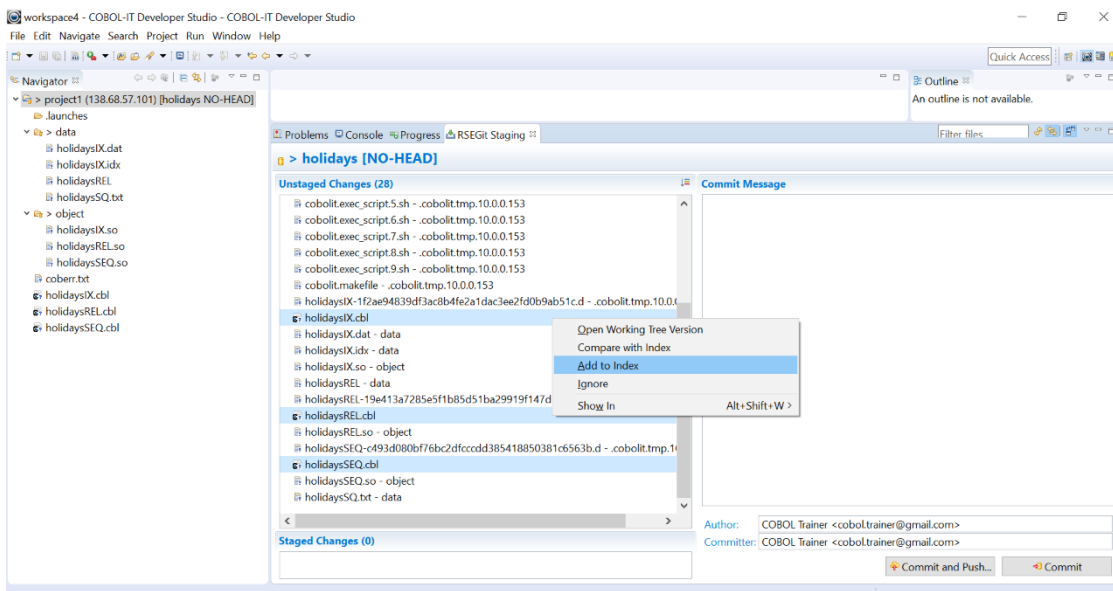
### Unstaged Changes

Before committing these files, we must first advance them to the “Staged Changes” area using the “Add to Index” function. To advance these files to the “Staged Changes” area, select the files, right-click, and select the “Add to Index” function from the dropdown menu.

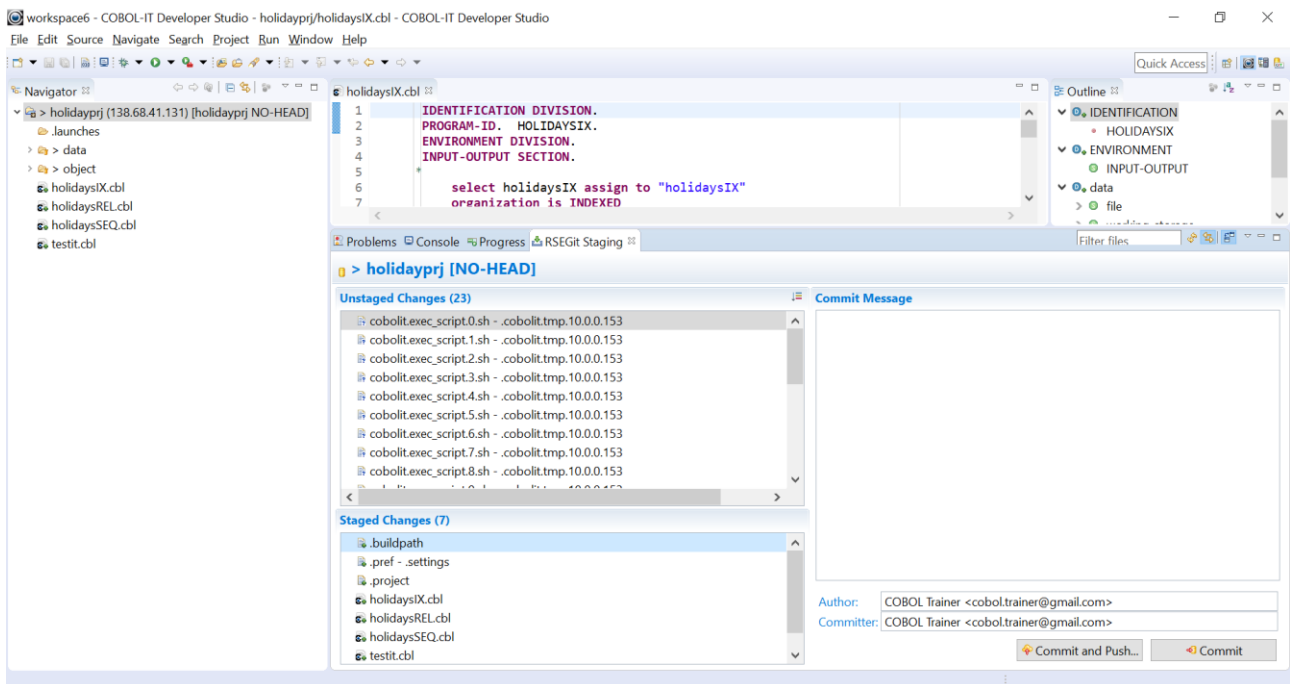


### ***Unstaged Changes>Add to Index***

Select the project files and the source files in the project folder and in the copy folder. Right-click, and select “Add to Index”.

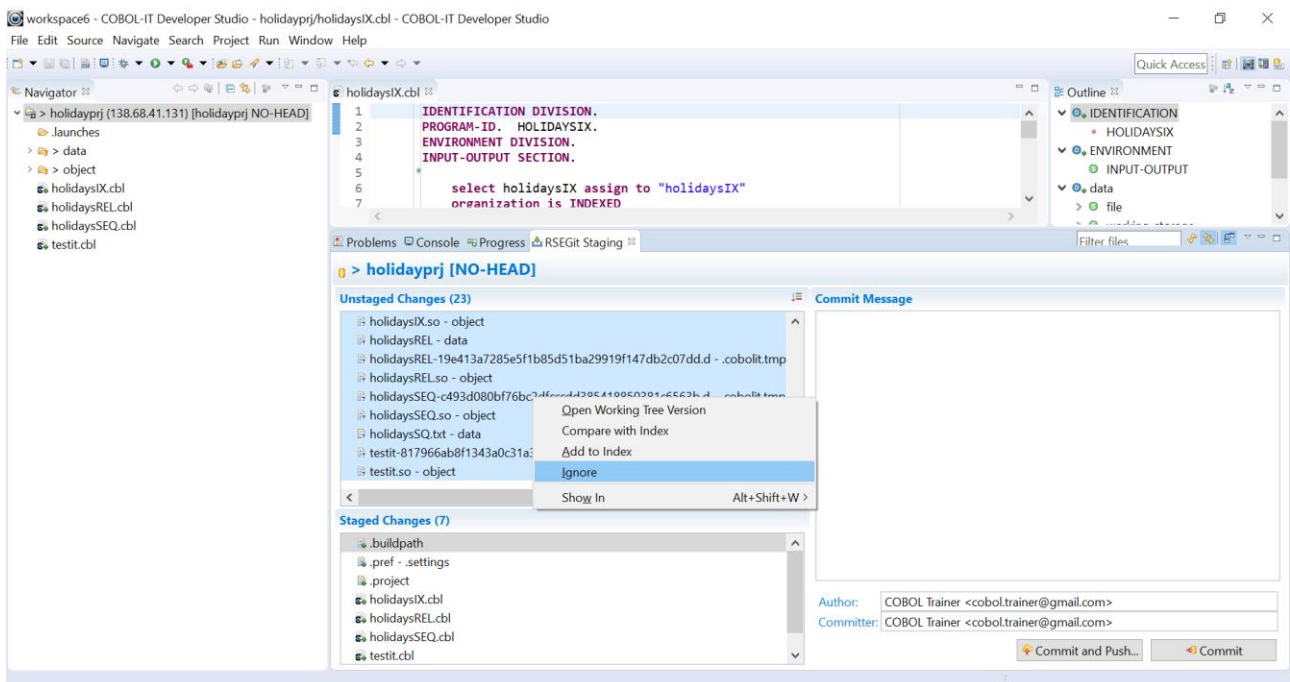


The selected files are transferred to the Staged Changes window.



### ***Unstaged Changes>Ignore***

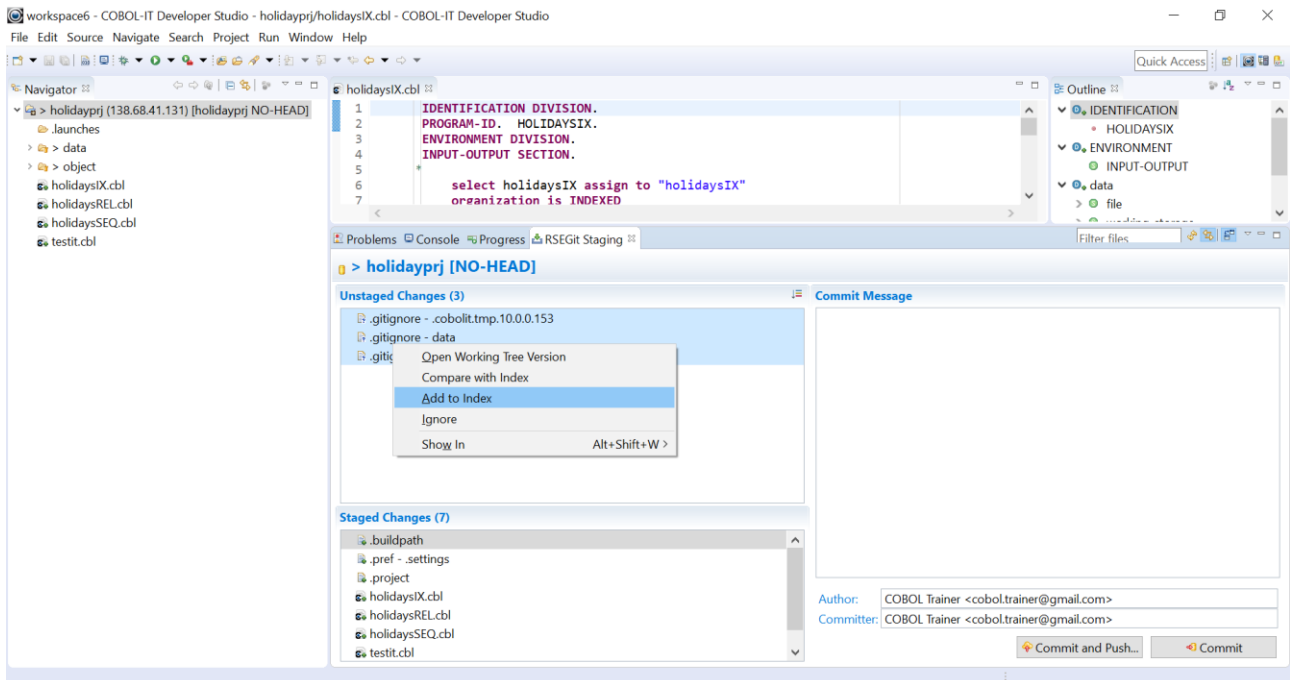
Select the remaining files. Right-click and select the “Ignore” function.  
This will cause the object and data folders to be labeled as “gitignore” folders.



### ***Unstaged Changes>gitignore folders>Add to Index***

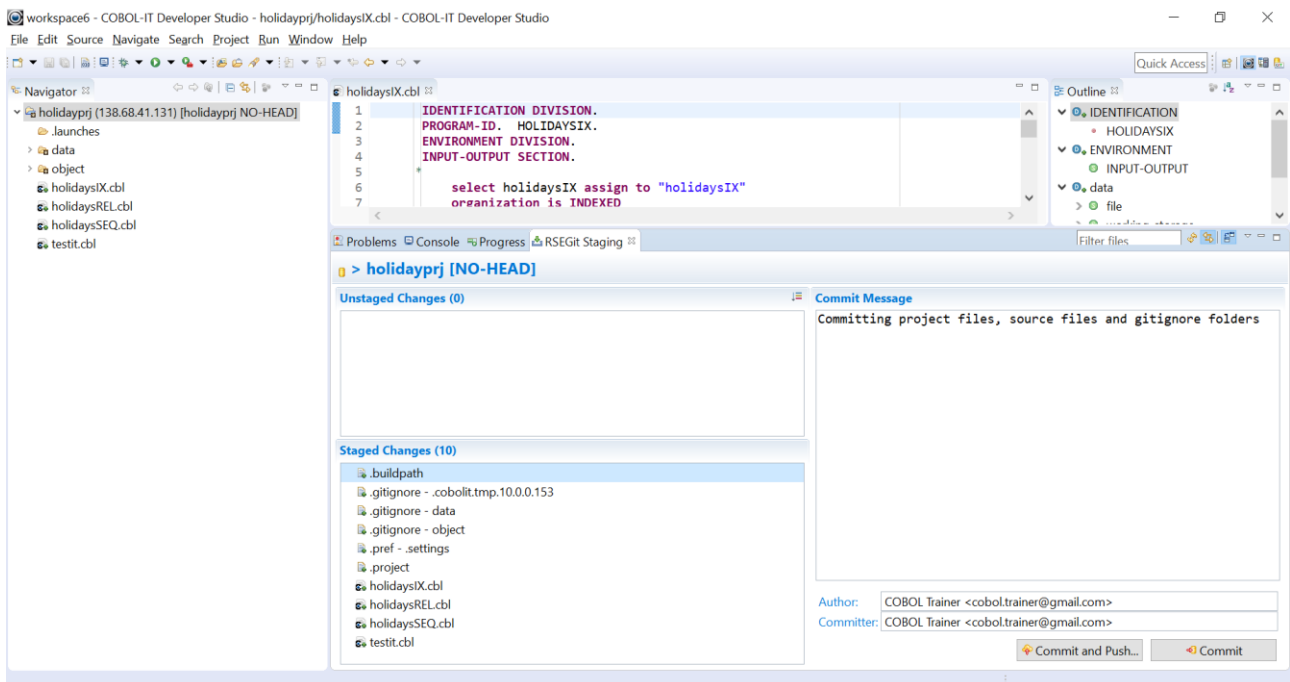
The data and object folders have been labeled as gitignore folders.

Select the gitignore folders, right-click and select the “Add to Index” function to move the gitignore folders from the “Unstaged Changes” window to the “Staged Changes” window.



## Staged Changes

The “Staged Changes” window now contains the project files, source files, and gitignore folders. To Commit, first enter a Commit Message, then press the “Commit” button.

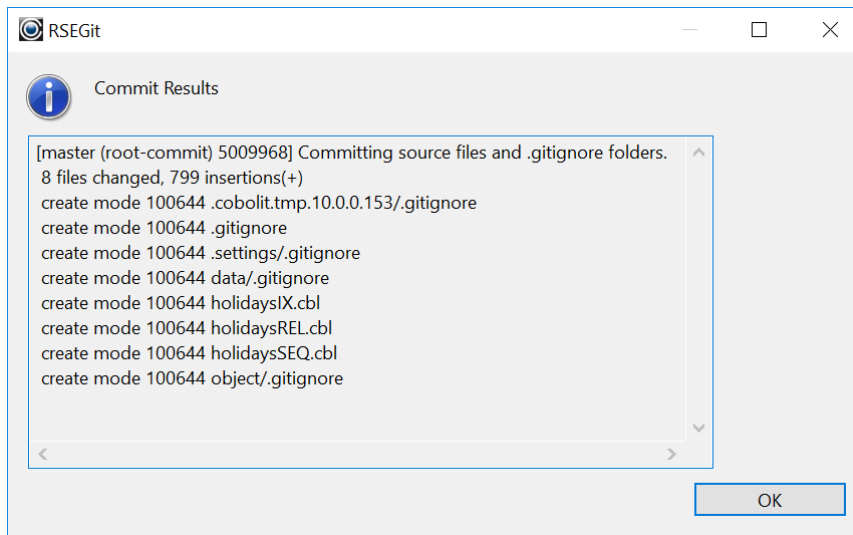


**Staged Changes>Commit Message & Commit button**

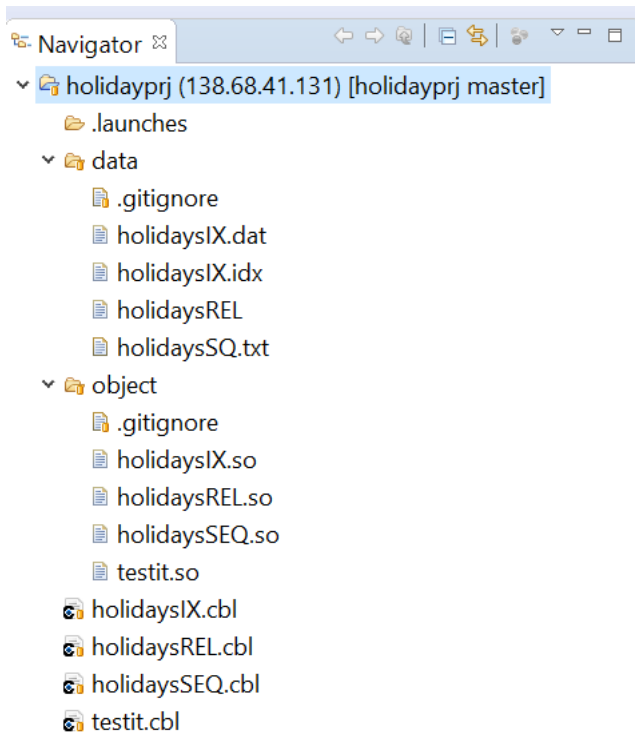
The Commit Message is associated with the Commit action, and preserved in the repository.

**Staged Changes>Commit Results**

RSEGit displays a Commit Results window summarizing the results of the Commit.

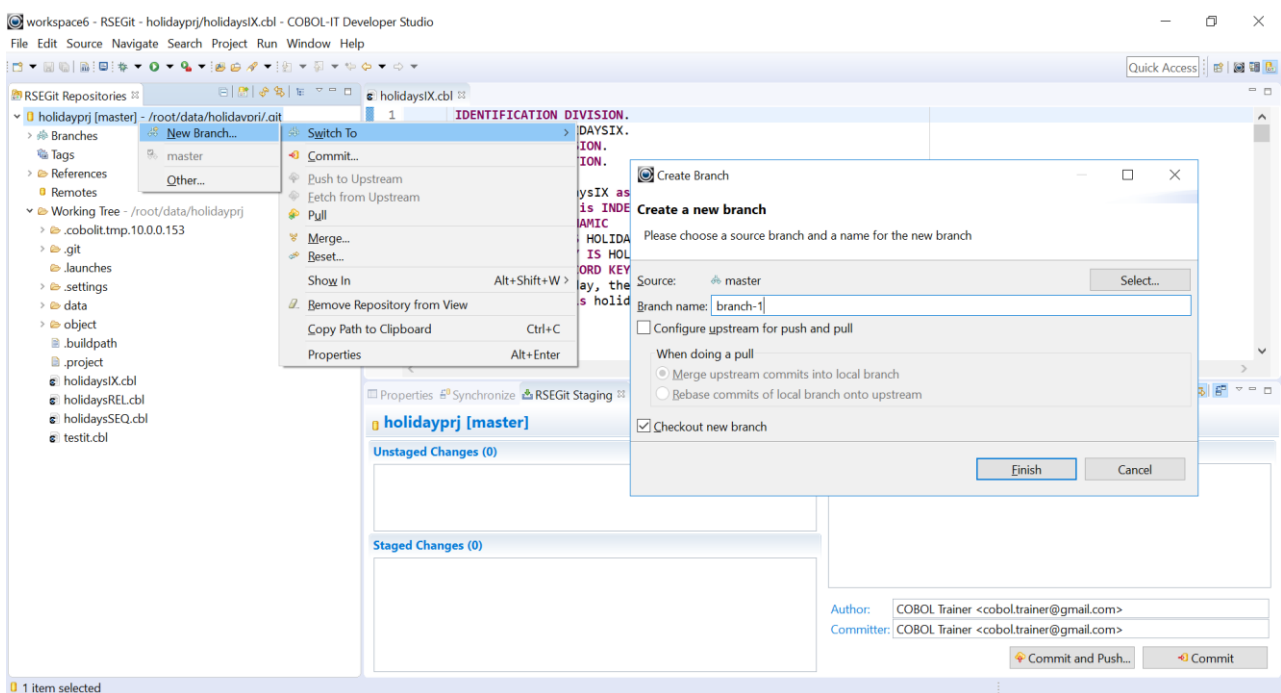
**The Navigator View of the COBOL Project>Tracked text**

The files are now tracked by RSEGit, and are marked with the “Tracked text” decoration.  
For details about Git Label Decorations, see Window>Preferences>Team>Git>Label Decorations.



## Create a New Branch

From the RSEGit Perspective, right-click on the RSEGit repository folder name, and select “Switch to...” from the dropdown windows, then select “New Branch” from the subsequent dropdown window. In the “Create Branch” dialog window, enter “branch-1” in the “Branch name” entry-field. Verify that the “Checkout new branch” checkbox is selected. Click “Finish”.

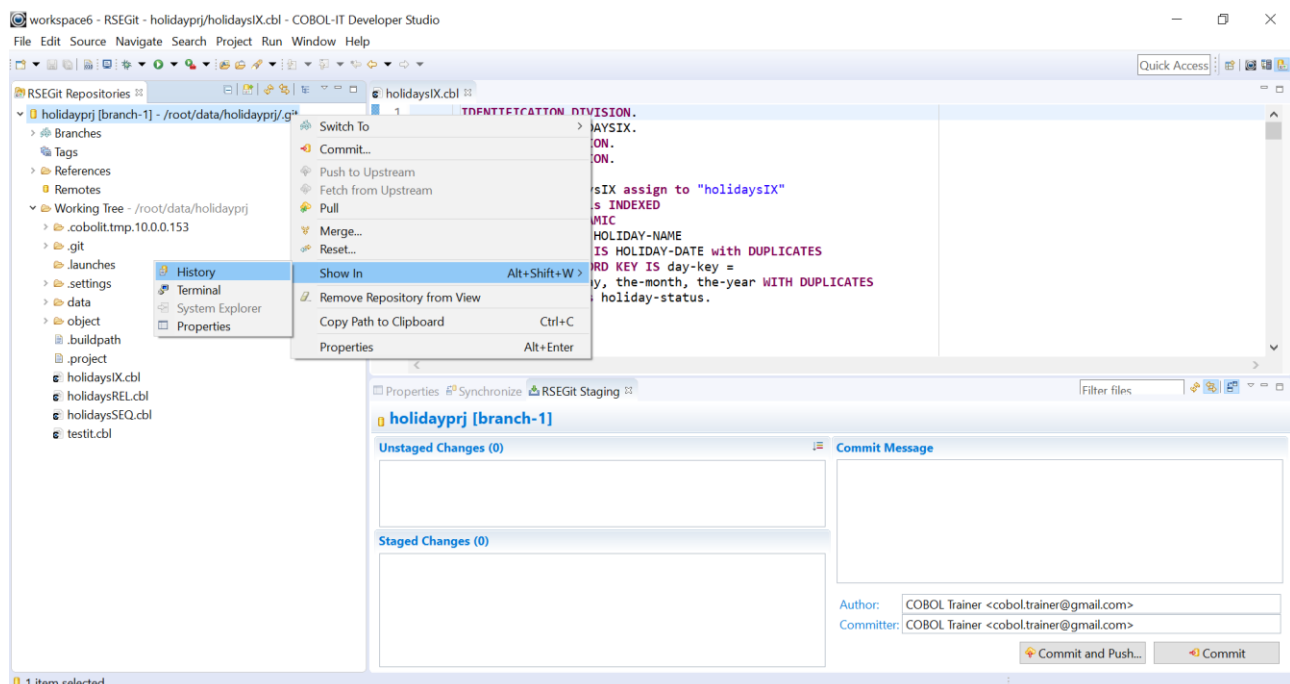


The new branch branch-1 has been created and checked out. Label decorations on your repository now include [branch-1]. You can work in this branch, and Commit to this branch, without affecting the master branch. You will notice that the [branch-1] label decoration is displayed on both the RSEGit Repositories View and in the Navigator View of the Developer Studio.

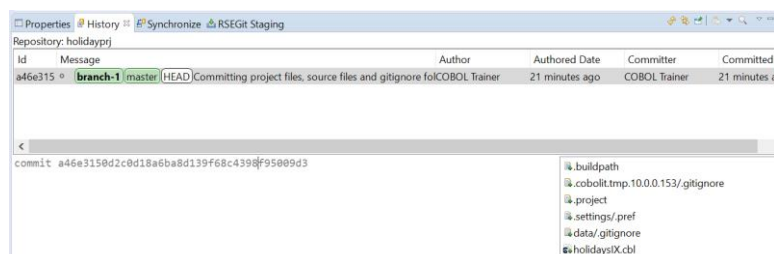
## Show In History

RSEGit> Show In>History

From the RSEGit Perspective, right-click on the RSEGit repository folder name, and select “Show In...” from the dropdown windows, then select “History” from the subsequent dropdown window.



This opens the History View in which information about the history of the branch is displayed.



## Importing Source Files into the Project

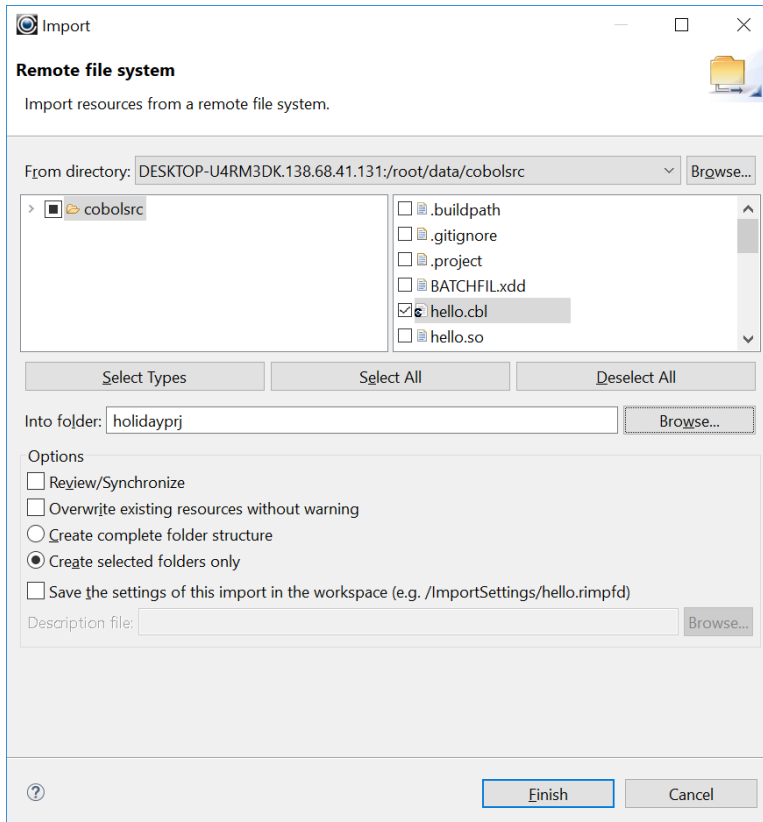
To import source files into the Project, enter the Remote System Explorer perspective, locate the source directory, right-click and select the “Import to project” function. In the Import dialog screen,





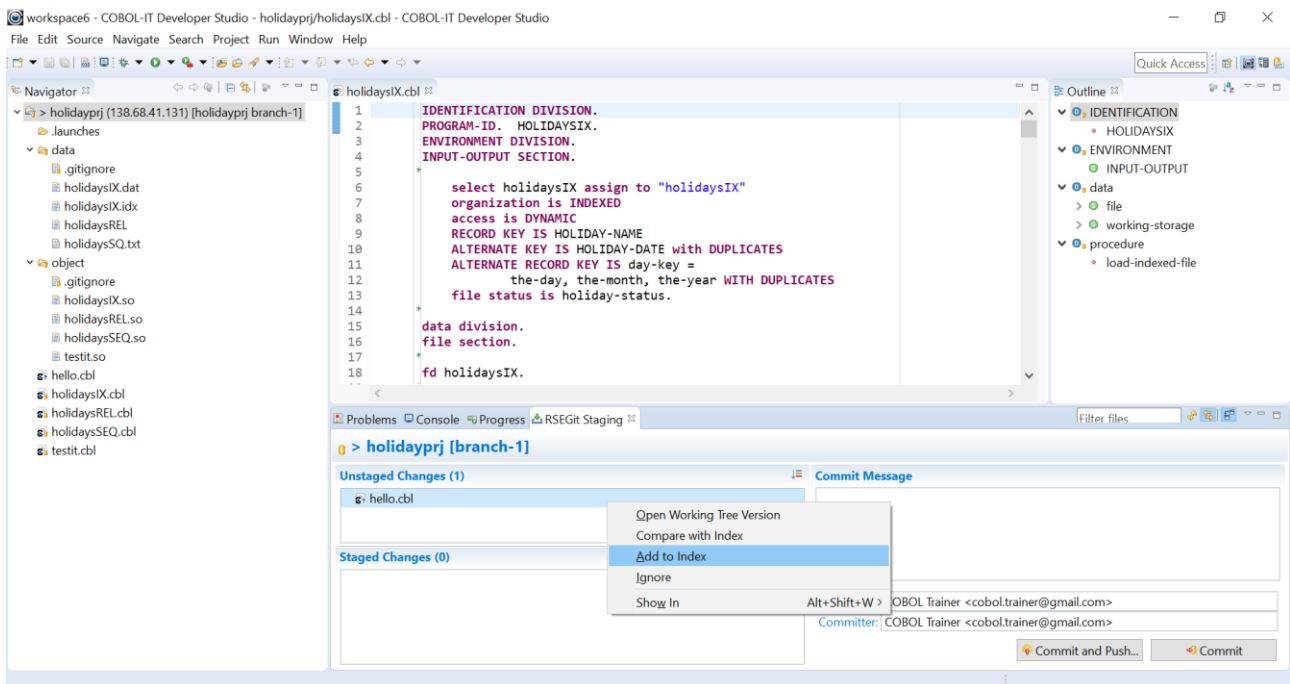
the selected source folder is displayed.

To fill the “Into folder” entry-field, click on the Browse button to the right of the entry-field, and in the “Import into Folder” dialog, select “project1” with its remote notation and Git label decorations. Click “OK” to return to the Import dialog screen. Click “Finish” to import the selected source into the project.

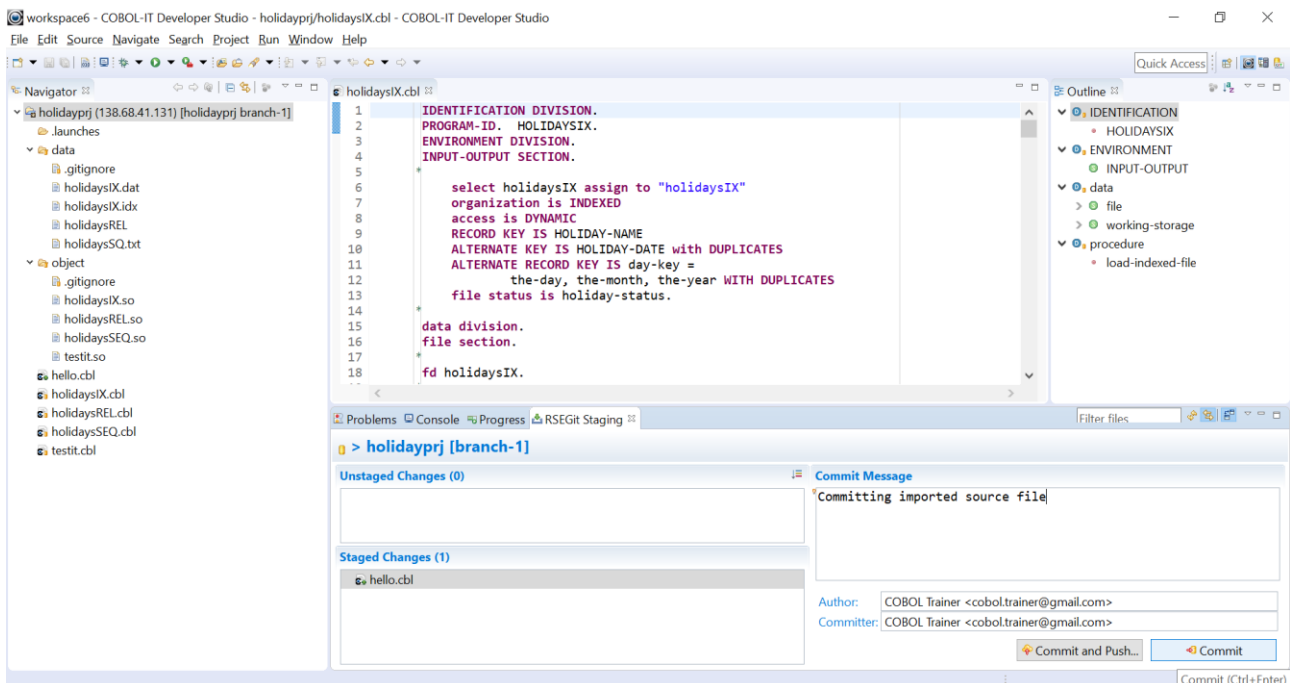


Return to the Developer Studio to see the file shown in the Project organization in the Navigator view. Note that files moved into the Project folder are automatically staged, and the source file is displayed as an “Unstaged Change”.

Select the file, right-click and select the “Add to Index” function from the dropdown menu. This will cause the source file to be viewed as a “Staged Change”.

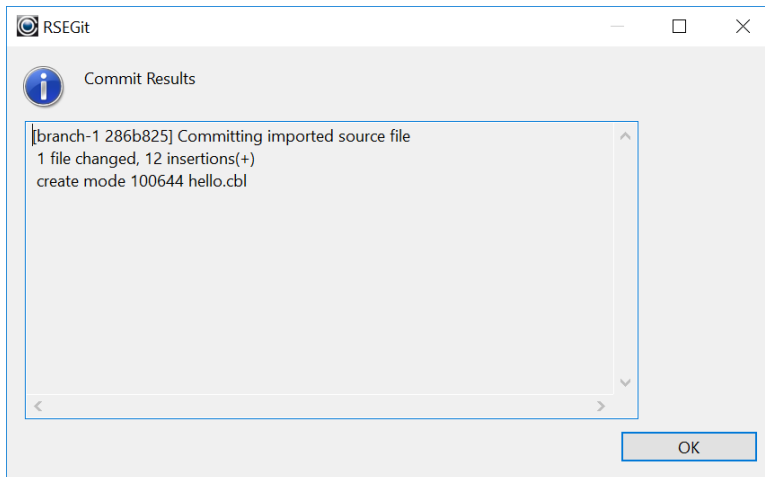


The file is moved from Unstaged Changes to Staged Changes.  
Enter a comment and click on the “Commit” button to commit the source file.



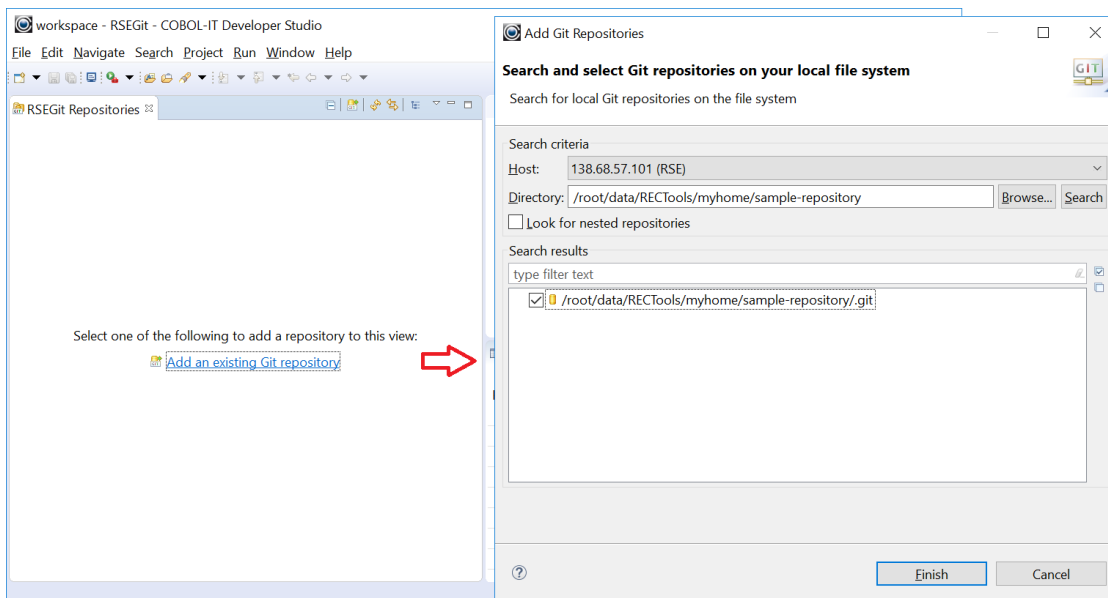
You will then see the Commit Results informative screen.



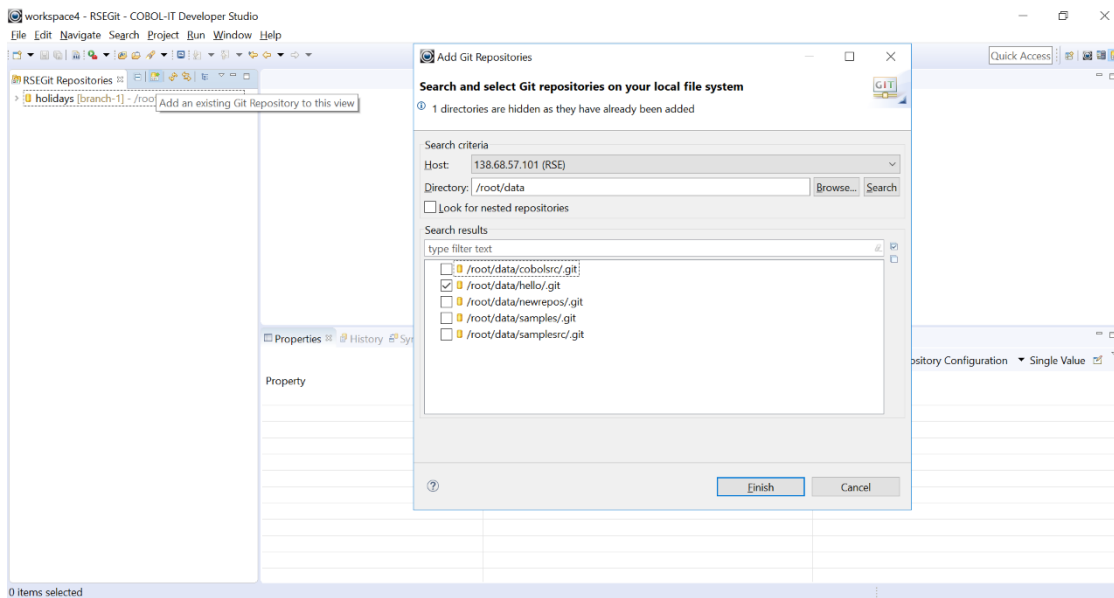


## Add an existing RSEGit Repository

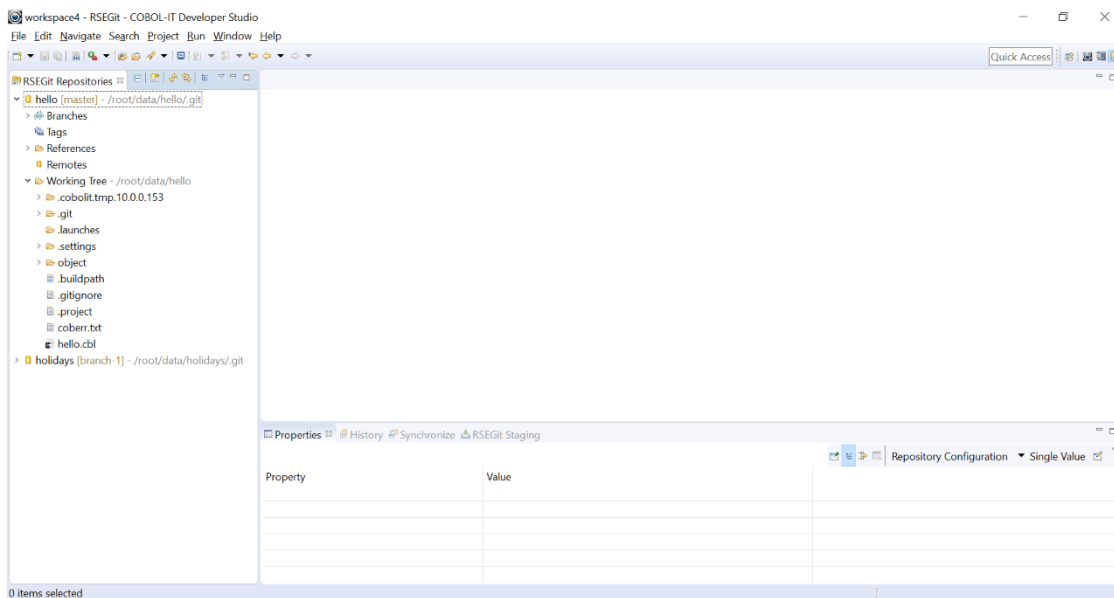
In the RSEGit Perspective, you can add existing RSEGit repositories to the RSEGit Repositories View. If your RSEGit Repositories view is empty, you will see a link titled “Add an existing Git Repository”.



If you have an RSEGit Repository displayed in the Repositories View, you may use the toolbar button with hint “Add an Existing Git Repository to this View. Set your Host to your remote machine, and then browse to a directory in which Git repositories reside. After selecting your directory, click on the Search button, and existing Git repositories are displayed in the Search results window. Select an existing Git repository to display it in the RSEGit Repositories View.



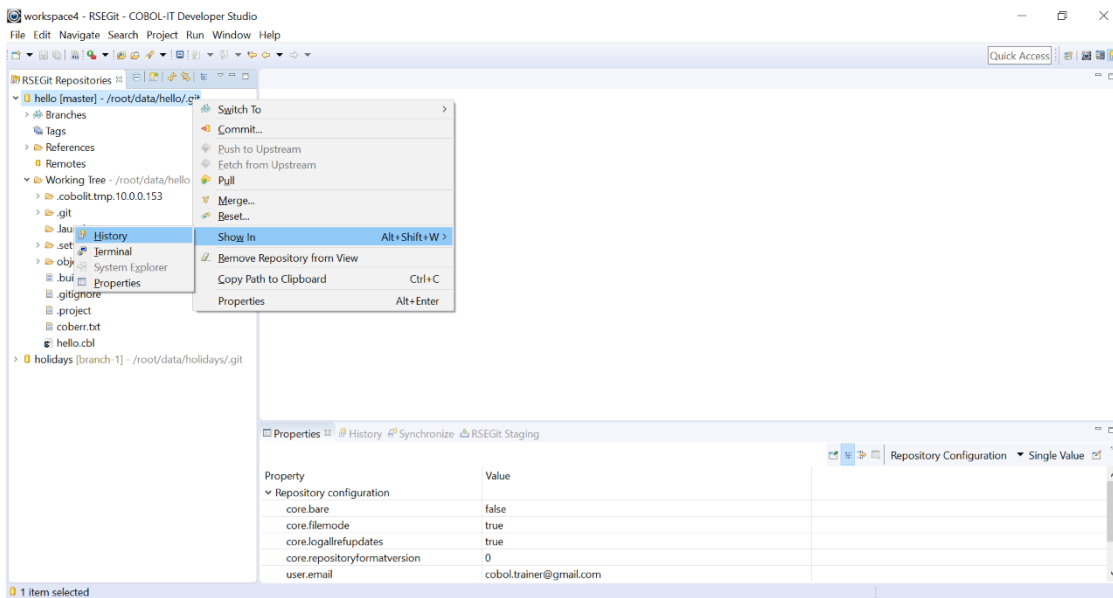
The selected repository displays in the RSEGit Repositories View.



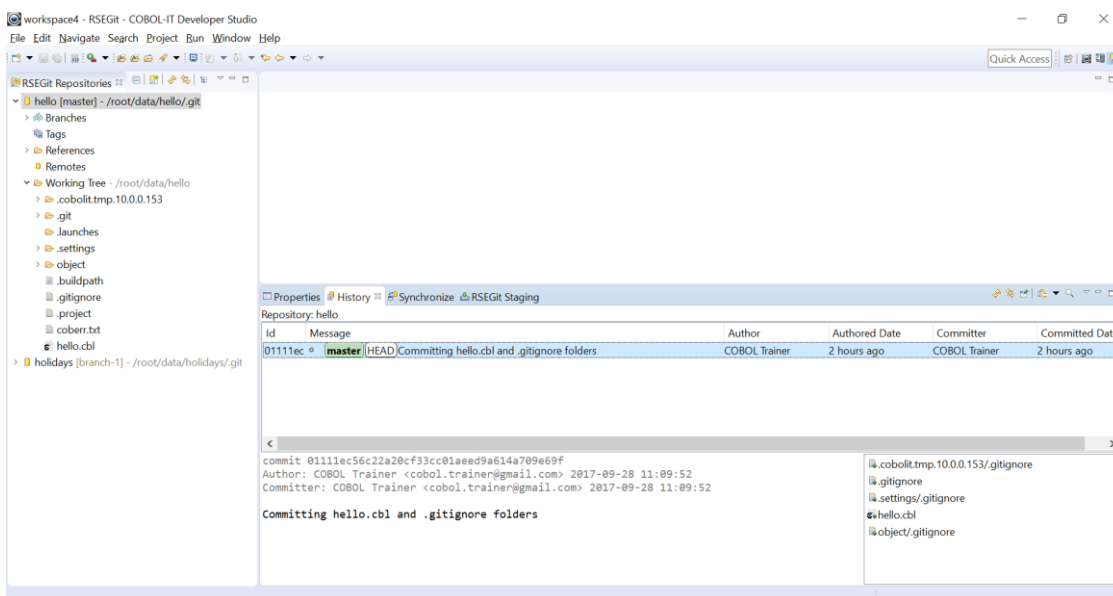
## Show In History

From the top-level dropdown menu, you now have access to a full array of Git functions. As an example, to Show the Git Repository in the History View, right-click on the top-level, and from the dropdown menu, select Show In > History.





This will open the History View for the Selected Repository.





## Using MyLyn for Task Management

Mylyn is the task and application lifecycle management (ALM) framework for Eclipse. With over 1 million downloads per month, Mylyn is the most popular IDE tool for ALM, and the task management tool of choice for developers around the World. 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.

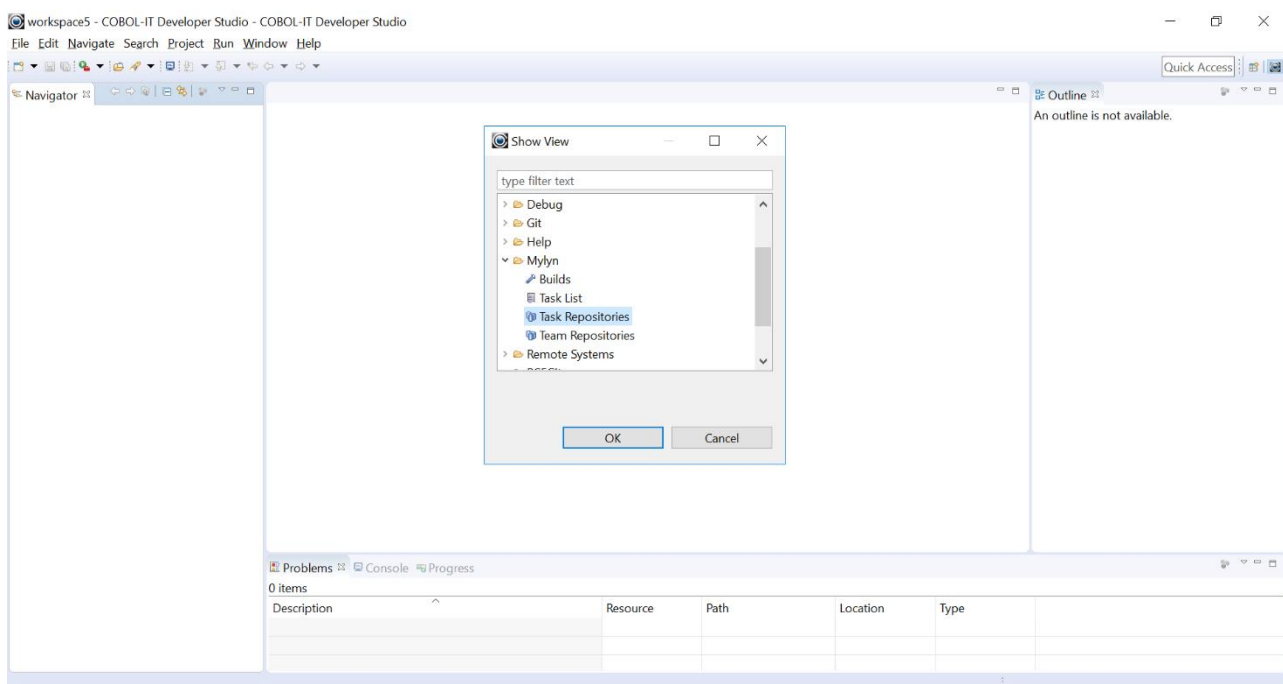
Note: Access the full Mylyn User Guide: [http://wiki.eclipse.org/Mylyn/User\\_Guide](http://wiki.eclipse.org/Mylyn/User_Guide)

In your Eclipse Installation Details, you will find references to the MyLyn Task List and a number of MyLyn connectors. The MyLyn Task List provides access to the MyLyn View. Connectors provide connectivity with issue management applications such as Git, and Bugzilla.

In the Developer Studio documentation, we will provide you with Getting Started scenarios for the use of MyLyn Task Management with a Remote Task Repository, such as Github, and with a Local Task Repository.

## Working with a Remote Task Repository

MyLyn is accessible through the Window>Show View menu. Drop down the menu under the Window selection on the Main Menubar, and select Show View... and then "Other" from the subsequent menu. This will open the "Show View" window. On the Show View menu, select Mylyn to expand the menu, and from the sub-menu, select Task Repositories.



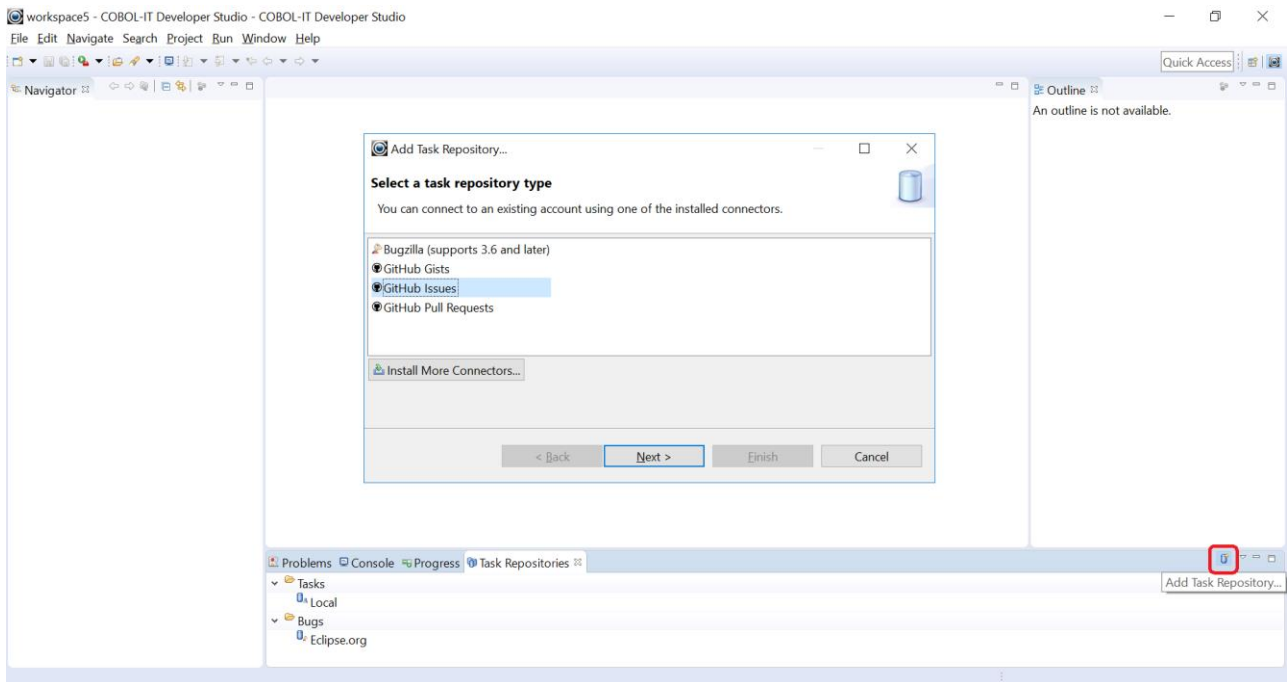
Click OK to open the Task Repositories View.





### Select a task repository type

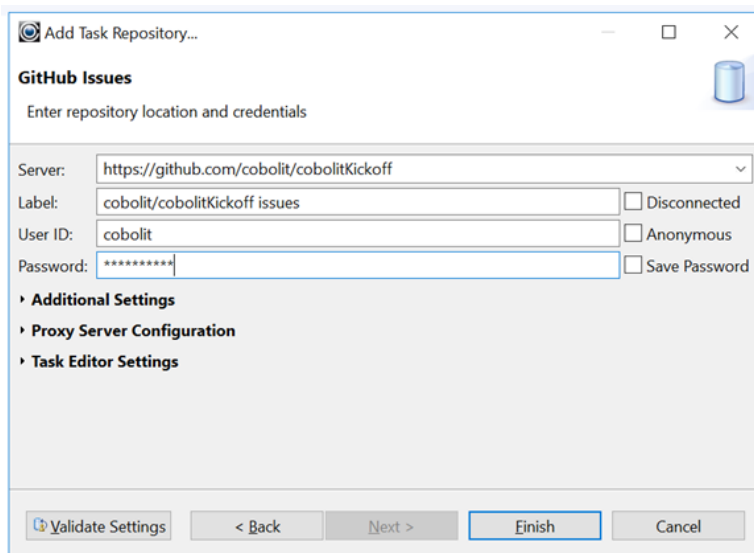
Click on the Add Task Repository button on the Task Repository View toolbar.



We then select Github issues, and click on the “Next” button to open the Github Issues window.

### Github Issues

On the Github Issues screen, we enter the Server, Label, and our User ID and Password, and click “Finish”.



In response to “Would you like to add a query?”, we click “Yes”.





## Enter query parameters

On the “Enter query parameters” screen, we enter a title, and then click on the “Finish” button.

**Edit Query**

**Enter query parameters**

Issue query settings

Title:

Status: ☒ Open ☒ Closed

Milestone:

Assigned to:

Mentioning:

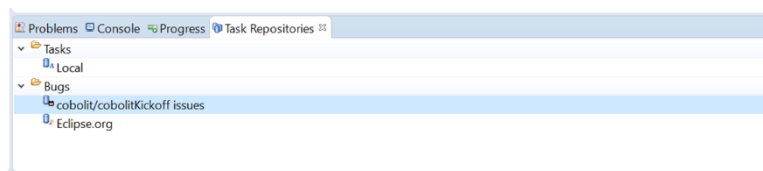
Labels:

- ☐ bug
- ☐ duplicate
- ☐ enhancement
- ☐ help wanted

We will then return to the Task Repository View.

## Task Repository View

Here, you can see that the Task Repository View has been updated and includes the remote “COBOLITKickoff Issues” repository. Next, we’ll open the Task List View.

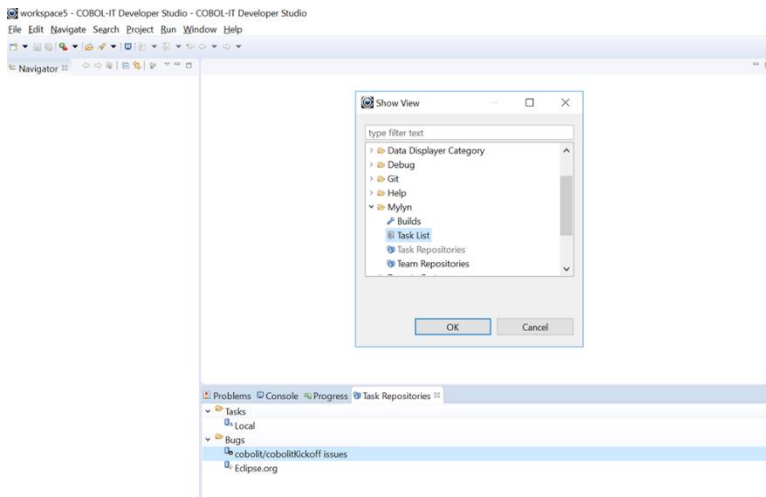


## Working with a Task List

Open the “Show View” window. On the Show View menu, select Mylyn to expand the menu, and from the sub-menu, select Task List.

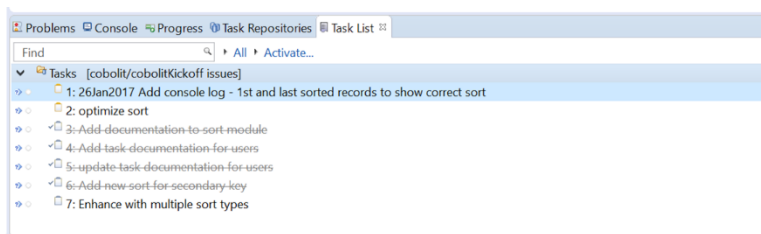






Click OK to open the Task List View.

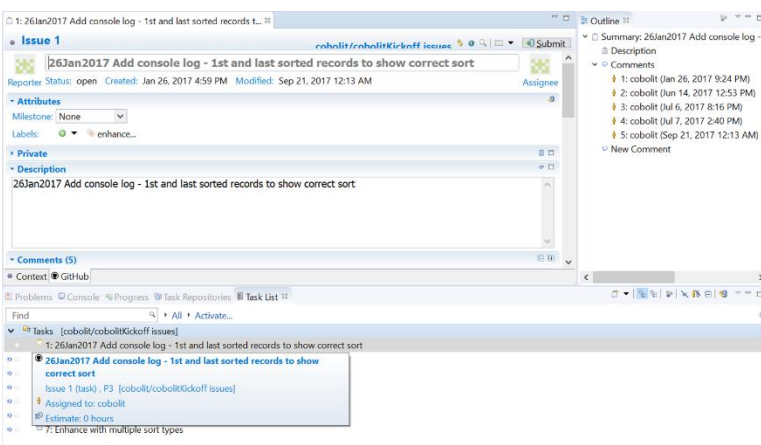
## Task List View



Double-click on the issue to expand it in the Task Editor.

## Handling an Existing Task

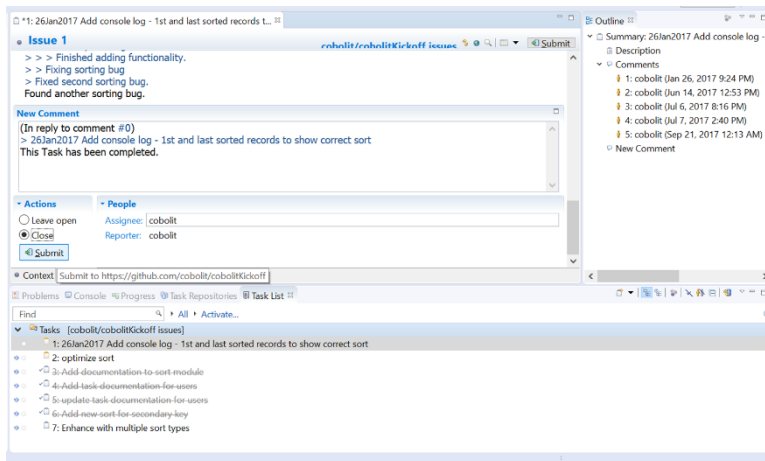
In the Task Editor you see all of the attributes of the issue. There is a scrollbar on the right which allows you to scroll down and see all of your options.





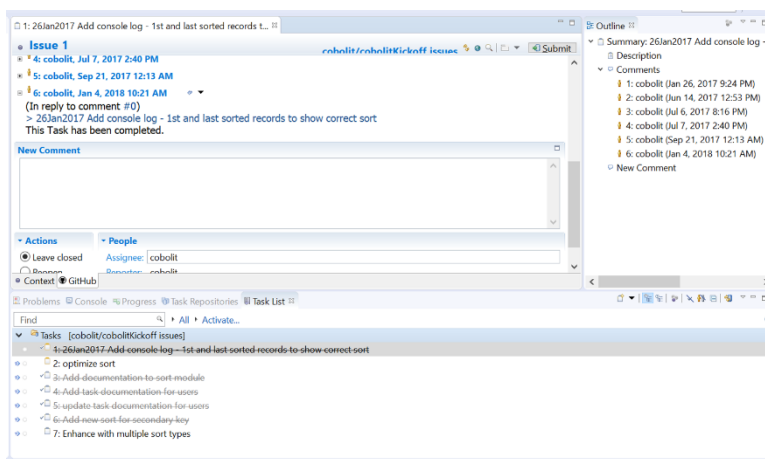
## Reply to the Task

Select Reply. Enter comments. To close an issue, select the “Close” action. Click on the Submit button to submit the update to Github.



## Submit Reply to the Task

The modifications have been submitted to github. The task is marked closed and summary updated.

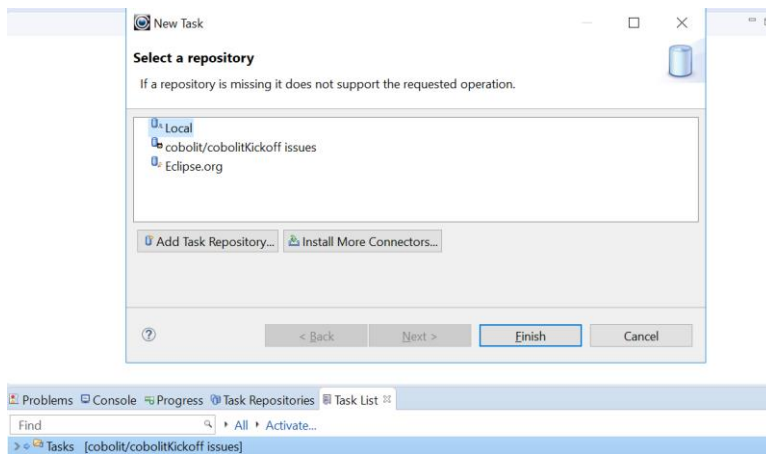


## Working with a Local Task Repository

### Create a new local task

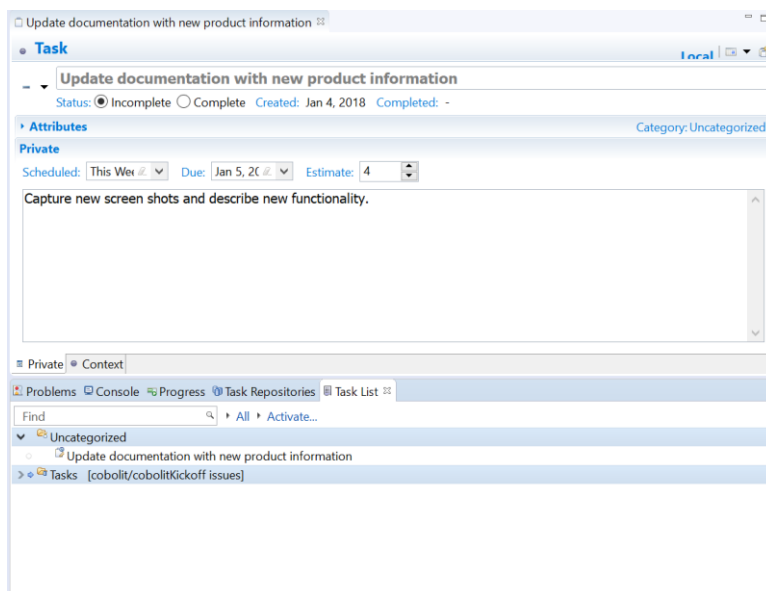
Mylyn also allows you to work with local tasks. Begin by selecting “New Task...” from the Task List View. Then from the Select a Repository window, select “Local”.





## Open a Local Task

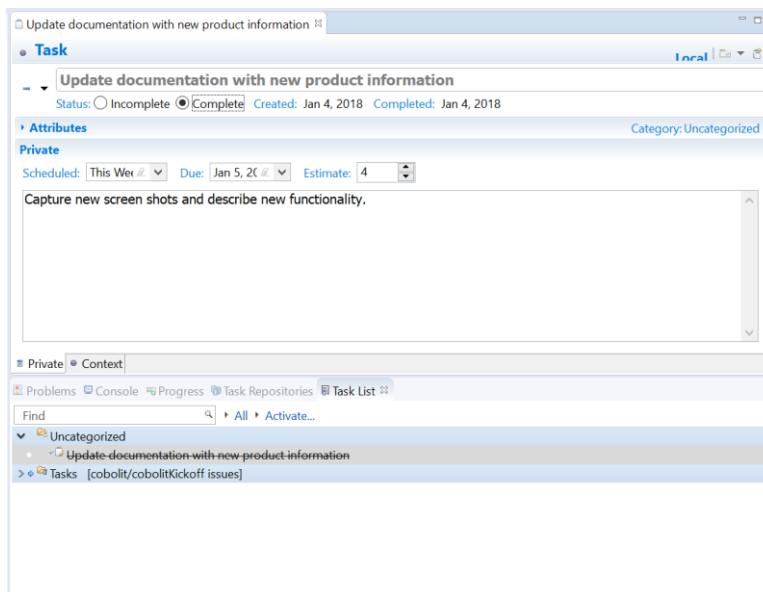
When the New task opens, assign it a name, enter some description, and save the task. You will see that it is added to your Uncategorized Task List.





### Close a Local Task

To close a task, mark it complete, and save it and you will see that it is marked with the strikethrough mark to indicate that it is closed.





**[www.cobol-it.com](http://www.cobol-it.com)**

June, 2020

