



COBOL-IT® Developer Studio Getting Started The Basics Version 2.0







# COBOL-IT Developer Studio- Getting Started The Basics

# **Contents**

ACKNOWLEDGMENT	4
COBOL-IT DEVELOPER STUDIO TOPICS	5
Introduction and License terms	
COBOL-IT Developer Studio License terms	
Dependencies	6
Dependencies and Comments	6
The COBOL-IT Developer Studio Distribution	6
USING THE COBOL-IT DEVELOPER STUDIO	7
Installing Developer Studio (Windows)	8
The Developer Studio Distribution	9
	10
Installing Updates to the Developer Studio	<b>10</b>
Administrator privileges required in some situations	10
	11
The Eclipse Platform	
Key Concepts	
The Workspace Launcher	
The COBOL-IT Developer Studio	13
Key Concepts	
Configuration Sottings	14
Window Preferences Ceneral Editors Text Editors	<b>14</b> 1/
Window>Preferences>General>Worksnace	14
Window>Preferences>COBOL>License file nath	
Window>Preferences>Run/Debug>Perspectives>	
Window>Preferences>COBOL>Compiler ( Compiler flags & Scripts )	
The File>New Wizerds	17
Key Concepts	17
File>New>COBOL Project	
Project>New>Folder	
Project>New>COBOL program	
Project>Properties>COBOL Properties ( Compiler flags & Scripts )	
Building and Running hello.cbl	
Project>Properties>COBOL Properties	
Standardize a Run Configuration	
Customize a Run Configuration	
The Run Configuration dialog window	
Set runtime environment variables on the Environment tab	
Compile, Run, Debug	
Key Concepts	
Clean, then Build	
The Clean Dialog Screen	





Key Concepts	
Managing Multiple Source Folders	
Close/Open a View	
Maximize/Restore a View	
Minimize/Restore a View	
Key Concepts	
Using "Views" in Eclipse	
FAQ	81
540	
Summary	
Dragging and dropping files and folders into a Project	
Run in Debug	
Run	77
Check the Compiler Console Window	77
Clean and Build	76
The COBOL Import Wizard	
Importing existing source code into a Project structure	72
Running the COBOL program in the Debugger Perspective	71
Run the COBOL program	71
Add the environment variables	71
Create a Run Configuration for project2>holidaysix.cbl	
The Compiler Console Window	
Clean and build project2	
Project>Properties>COBOL	
Project>Properties>Resource	
Eclipse artifacts are created in the existing directory	
New Project Using Existing Source	66
Using existing source code in its current location	
ADDITIONAL DEVELOPMENT SCENARIOS	66
Summarizing key features of the Developer Studio	65
Correcting compiler errors from the Compiler Console View	
Correcting compiler errors from the Problems View	
Toggle Perspectives to re-enter the Developer Studio Perspective	
Running in the Debugger Perspective	61
Running the COBOL program	
Open With>Text Editor to view text files	
Cical, Dund, and Rebuild a single program	
Clean Build and Rebuild a single program	





# 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:**

The Lawn 22-30 Old Bath Road Newbury, Berkshire, RG14 1QN United Kingdom Tel: +44-0-1635-565-200





# **COBOL-IT Developer Studio Topics**

# **Introduction and License terms**

This document describes how to install and how to use the **COBOL-IT Developer Studio**, which 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<sup>®</sup> is wholly owned by COBOL-IT. Any unauthorized reproduction of the software without the express written consent of COBOLIT is prohibited.

COBOL-IT Corporate Headquarters are located at:

The Lawn 22-30 Old Bath Road Newbury, Berkshire, RG14 1QN United Kingdom Tel: +44-0-1635-565-200

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.





# 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 www.microsoft.com.
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 <u>sales@cobol-it.com</u> .
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.

# **Using the COBOL-IT Developer Studio**

This document describes how to install and how to use the **COBOL-IT Developer Studio**. **COBOL-IT Developer Studio** is a COBOL development workshop based on the Eclipse environment.

This IDE includes a project manager and a full screen editor dedicated to COBOL language with colorized syntax and an interface to the COBOL debugger. It may be customized according to the specific needs of each company. The Eclipse environment offers great flexibility through its wide range of plug-ins.

The Developer Studio is the Eclipse-based COBOL-IT Development Environment, providing a framework for using the Compiler, Runtime and Debugger in a Development environment with a feature-rich COBOL Code Editor.

**COBOL-IT Developer Studio** enables COBOL developers to more easily maintain and enhance their applications by offering them the entire range of possibilities and power of Eclipse.

This chapter guides the user through a practical exercise that includes installing the Developer Studio, and launching a workspace.





# Installing Developer Studio (Windows)

COBOL-IT Developer Studio Windows distributions are provided in a zipped folder file format. Distributions are provided for the Kepler Version of Eclipse (Java 6, 7), as well as for the Neon version of Eclipse (Java 8).

Downloadable distributions for the Developer Studio version 2.0.0 are available in both 32- and 64bit formats for Windows and Linux operating environments, with builds done with both the Kepler and Neon versions of Eclipse. Linux distributions are provided in gzipped tar formats. Windows distributions are provided in zipped formats.

The COBOL-IT Developer Studio is downloadable from COBOL-IT Online, with access provided by your Sales Representative. Your login and password will give you access to the Downloads Screen on COBOL-IT Online.

In the Active Authorizations Window of the Downloads screen, select "Cobol IT Developer Studio". Active Authorizations are the Products you are currently authorized to Download. Selecting "COBOL-IT Developer Studio" displays a list of the Products you may download.

From the list, select IDE Windows with jre.

After selecting "IDE Windows with jre", you will advance to the COBOL-IT Developer Studio Download Screen.

#### For Windows Distributions:

Right-click on the COBOL-IT Developer Studio zipped archive and select "Unzip to…". Then browse to the folder in which you wish to install your Developer Studio.

The COBOL-IT Developer Studio is an application called "CDS.exe". Double-click on "CDS" to launch the Developer Studio.

#### For Linux Distributions:

From the command line, unzip/untar the distribution archive, and install it in a directory on your Linux Server.

>tar zxpf devstudio-neon-2.0.0-linux.gtk.x86\_64.tar.gz -C /home/cobolit/devstudio

The Linux-based COBOL-IT Developer Studio is an application called "CDS". Run "CDS" to launch the Developer Studio.

The Developer Studio is licensed with the COBOL-IT Developer Studio license file which is located by default in your COBOL-IT installation directory. Licenses are product- and platform specific. In addition to the Developer Studio license, usage of the Developer Studio requires a license for the COBOL-IT Compiler Suite for all platforms on which you wish to compile and run COBOL programs.





# The Developer Studio Distribution

F Home Share	per Studio					- 0	× ^ (
Pin to Quick Copy Paste	Cut Copy pa	th nortcut to v to v	New item •	Properties	n * Select all		
$\leftarrow \rightarrow \checkmark \uparrow \downarrow \rightarrow$	This PC > (	Organize	oper Studio >	open	・ じ Search Develo	per Studio	,o
Desktop	* ^	Name	Date m	odified T	VDe	Size	
Downloads	*		7 (27 (26				
Documents	*	footures	7/27/20	17 10:51 AM	ile folder		
Pictures	*	p2	7/27/20	017 10:51 AM F	ile folder		
CompilerSuite		plugins	7/27/20	017 10:51 AM	ile folder		
GettingStarted		📜 readme	7/27/20	017 10:51 AM	ile folder		
project1		.edipseproduct	1/3/201	17 1:14 PM E	CLIPSEPRODUCT File	1 KB	
project1		artifacts	7/21/20	017 2:45 PM	ML Document	108 KB	
project i		CDS	7/21/20	017 2:44 PM A	pplication	312 KB	
Dropbox		CDS	7/21/20	017 2:45 PM C	onfiguration settings	1 KB	
a OneDrive		eclipsec	7/21/20	017 2:44 PM A	pplication	24 KB	
		e epi-viu	11/9/20	16 3:34 AM F	ITML File	16 KB	
S This PC		• houce	11/5/20	710 3:34 AM F	ITML FIE	7 ND	
🔁 Desktop							
📔 Documents							
漙 Downloads							
Music							
💺 Pictures							
H Videos							
🐛 OS (C:)							
12 items	~					8	

Validate the Installation, by checking the license.

To validate the install, select Window, from the Main Menubar, and then select "Preferences" from the dropdown menu. In the Preferences dialog screen, select "COBOL" from the panel on the left. You will now need to browse to locate your COBOL-IT license file. The Developer Studio will attempt to validate licenses located in the installation directory with a name of citlicense.xml, or a name beginning with "citlicense-" and with the ".xml" suffix.

As an examples, click on the Browse button, navigate to the installation directory, and select "citlicense.xml". The expiration date of your license, and the products and platforms supported will be displayed. Verify that the Compiler Suite and Developer Studio are licensed on your platform.





type filter text	COBOL	ф <b>т</b>	⇔ ▼	•
<ul> <li>General</li> <li>General</li> <li>GOBQL</li> <li>Dynamic Languages</li> <li>Help</li> <li>Install/Update</li> <li>Mylyn</li> <li>Remote Systems</li> <li>Run/Debug</li> <li>Team</li> <li>Terminal</li> </ul>	General COBOL settings License file path: C\CobohCobolIT/citlicense.xml Registered to: COBOL-IT Demo Rue de la Caserne, 45 B-1000 Bruxelles Belgique Subscription: Distributor - Personal use Only Expires: 2017-12-31 Platforms: ZLINUX, HPUX, AIX, WINDOWS, LINUX, SOLARIS Products: SQL, SYNTAX-ONLY, COMPILER, IDE, RUNTIME, REPORT-WRITER, EBCDIC Created by: COBOL-IT Demo Rue de la Caserne, 45 B-lo00 Bruxelles Belgique	Δp	}rows	P
0		~		

Click OK to return to the Developer Studio.

# Installing Updates to the Developer Studio

# Administrator privileges required in some situations

In Windows, if the Developer Studio is installed under either the C:\Program Files or C:\Program Files (x86) folders on your Windows system, then you must have Administrator privileges in order to perform Developer Studio updates. This is because Windows considers the C:\Program Files and C:\Program Files (x86) folders to be restricted directories.





# To install an update to the Developer Studio

Select Help/Check for Updates from the Main Menubar.

	© w	orkspace - COBOL-IT Developer Studio - (	COBOL-IT De	eveloper Stu	ıdio		_		×
ł	File E	dit Navigate Search Project Run Win	dow Help						
Ð	0	Help Contents					Quick	Access	1
9	*	Search					🗄 Outli	ne 🛙	- 8
	đ	Show Active Keybindings Ctrl Tips and Tricks Report Bug or Enhancement Cheat Sheets Check for Updat	+Shift+L				An outli availabl	ine is no <sup>t</sup> e.	t .
	Ŷ	Check for Updates							
	-	Install New Software							
	1	Installation Details							
	۲	About COBOL-IT Developer Studio							
		🗈 Problems 😂 💷 Con	isole 🖷 Pro	gress				<u>5</u> 9	~
1		0 items							
L		Description	^			Resource		Path	
L		<							>

Eclipse is programmed to check the plug-ins you have installed for updates.

On the Available Updates dialog screen, click "Select All", and then click on the "Next>" button to continue.

Review and confirm the updates.

If any updates are downloaded, you will be required to review and accept the terms of the Eclipse license agreement.

Select the radio button titled "I accept the terms of the license agreement.". Click on the Finish button to begin the download and install of the updates.

Wait, while your software updates. You may receive a Security Warning that the software contains unsigned content. Click "OK". When you are prompted to restart Eclipse, click "Restart Now". To verify that your update has been installed, click on Help, then click on Installation Details in the dropdown menu.



© COB	OL-IT (	COBOL-IT Develope The	r Studio– G Basics	etting Started	Version 2.
<ul> <li>workspace - COBOL-IT Developer Studic File Edit Navigate Search Project Run M Search</li> <li>Help Contents</li> <li>Search</li> <li>Show Active Keybindings</li> <li>Tips and Tricks</li> <li>Report Bug or Enhancement</li> <li>Check for Updates</li> <li>Install New Software</li> <li>Installation Details</li> <li>About COBOL-IT Developer Studio</li> </ul>	<ul> <li>COBOL-IT Developer Studio</li> <li>Window Help</li> <li>Ctrl+Shift+L</li> <li>About COBOL-IT Developer Studio</li> <li>COBOL-IT Developer</li> </ul>	- C X Quick Access : B @ COBOL-IT Developer Studio Install	ation Details		- 🗆 X
🖁 Problems 🕮 💷 (	⑦ Installation Details Console ➡ Progress	Installed Software Installation Histo	ry Features Plug-ins Con	Id	Provider
0 items Description <	*	<ul> <li>✓ GC0BOL-IT Developer Studio</li> <li>GCode Coverage</li> <li>&gt; Go Data Display</li> <li>&gt; Be DevStudio</li> <li>&gt; Eclipse Git Feam Provider</li> <li>&gt; Be Eclipse RCP</li> <li>&gt; B/RE Feature</li> <li>© Licensing</li> <li>© Profiler</li> </ul>	2.0.0.201712051120 2.0.0.201712051120 2.0.0.201712051120 2.0.0.201712051120 3.0.3.201309161630-r 4.3.1.v20130911-1000 4.3.1.v20130911-1000 1.0.0 1.9.0.201709150700 2.0.0.201712051120	com.cobolit.eclipse.product com.cobolit.eclipse.coverage.feature.feature.gro. com.cobolit.eclipse.display.data.feature.feature com.cobolit.eclipse.feature.feature.group org.eclipse.egit.feature.group org.eclipse.platform.feature.group org.eclipse.rcp.feature.group jre6.feature.feature.group com.cobolit.eclipse.licensing.feature.feature.gro. com.cobolit.eclipse.profiler.jre6.feature.feature.grout.group.	COBOL IT . COBOL IT COBOL IT Eclipse EGit Eclipse.org Eclipse.org COBOL IT COBOL IT COBOL IT
		< 	Update	Uninstall Properties	Close

The COBOL-IT Developer Studio Installation Details window contains Version Information about COBOL-IT Developer Studio, and its dependencies in the Eclipse environment.

# The Eclipse Platform

# Key Concepts

The Eclipse Platform stores your projects in a folder called a workspace.

When opened, Eclipse prompts the user to indicate where they wish to launch their workspace.

Eclipse guides the user from the "Workspace Launcher" screen through the "Welcome" screen, to the Developer Studio IDE.

Users of the Developer Studio IDE should begin by de-selecting "Build Automatically" on the dropdown menu under the Project button on the main mainubar.

## The Workspace Launcher

Enter a path for the Workspace, and click "OK".







# The COBOL-IT Developer Studio

### Key Concepts

This chapter guides the user through the installation of the COBOL-IT Developer Studio and includes remarks about the Eclipse IDE. Then, there is a review of best practices for using the COBOL-IT Developer Studio.

Best practices for using the COBOL-IT Developer Studio are:

- Enter the Window>Preferences interface and configure the Development Studio. Minimal configurations can be made in Window>Preferences>General>Editors>Text Editor, Window>Preferences>General>Workspace and Window>Preferences>Run/Debug>Perspectives.
- 2- Use the File>New>Project>COBOL Project Wizard to create a new project.
- 3- Use the File>New>COBOL Program Wizard to create a new COBOL program in the project folder.
- 4- Enter the Project>Properties>Compiler Options interface to set the compiler flags in your project.
- 5- Create a runtime configuration for your COBOL program.
- 6- Clean, and Build the Project.
- 7- Review the output from the Build in the Compiler Console Window, and possibly the Problems window. The Problems window will display compiler errors detected in a clickable interface that returns control to the line on which the compiler error was detected.
- 8- Run the program.
- 9- Run the program in the debugger.

We have also included reference chapters for :

Additional development scenarios, including:

- a. Using existing source code in its current location
- b. Importing existing source code into a New Project structure
- c. Dragging and dropping files and folders from the Windows explorer into the Project

For reference material on using the COBOL-IT Debugger, see the related document :

• Getting Started with the Developer\_Studio Debugger Perspective





For reference material on using the Remote System Perspective, see the related document:

- Getting Started with the Developer Studio
  - Remote System Perspective

With these guidelines, we hope that your experience of Getting Started with the COBOL-IT Developer Studio will be rewarding. The COBOL-IT Technical Services team welcomes your feedback. Please submit comments and suggestions to <u>techsvcs@cobol-it.com</u>.

# **Configuration Settings**

Configuring your development environment using the Window>Preferences interfaces causes the settings to be applied at the Workspace level. Settings applied at the Workspace level are applied automatically to all Projects in a Workspace.

### Window>Preferences>General>Editors>Text Editors

Configure behaviors of the Text Editor. Select checkbox to enable the use of :

Show line numbers

Causes line numbers to print in columns 1-6

Note- The COBOL Code Editor derives its Tab behaviors from the Window>Preferences>COBOL>Editor dialog screen.

Preferences				$\times$
type filter text	Text Editors		⇔ ▼ ⇔	• •
<ul> <li>✓ General</li> <li>&gt; Appearance</li> <li>Compare/Patch</li> <li>Content Types</li> </ul>	Some editors may not honor all See <u>'Colors and Fonts'</u> to config	of these settings. ure the font.		^
<ul> <li>Editors         <ul> <li>Autosave</li> <li>File Associations</li> <li>Text Editors</li> <li>Globalization</li> <li>Keys</li> </ul> </li> <li>Network Connections         <ul> <li>Notifications</li> </ul> </li> </ul>	Undo history size: Displayed tab width: ☐ Insert spaces for tabs ☑ Highlight current line ☐ Show print margin Print margin column: ☑ Show line numbers ☑ Show line numbers	200 4 80		
Perspectives Search > Security > Startup and Shutdown UI Responsiveness Monitoring Web Browser > Workspace > COBOL > Dynamic Languages > Help > Install/Update > Mylyn > Remote Systems	<ul> <li>Show range indicator</li> <li>Show whitespace characters (</li> <li>Show affordance in hover on</li> <li>When mouse moved into hover:</li> <li>✓ Enable drag and drog of text</li> <li>✓ Warn before editing a derivee</li> <li>✓ Smart caret positioning at line</li> <li>Appearance color options:</li> <li>Line number foreground</li> <li>Current line highlight</li> <li>Print margin</li> <li>Eind scope</li> </ul>	configure visibility) how to make it stick Enrich after delay d file e start and end	5y ~	
<ul> <li>Run/Debug</li> <li>Team</li> <li>Terminal</li> </ul>	Selection foreground color Selection background color Background color Foreground color Hyperlink	OK	Cancel	~





## Window>Preferences>General>Workspace

Configure behaviors associated with the Workspace. **De-select checkbox to disable the use of :** 

Build automatically De-select Build automatically

#### Select checkboxes to enable the use of:

Refresh using native hooks or polling Refresh on access Save automatically before build Refresh view of project files using native hooks Refresh view of project files on access Causes unsaved changes to be saved before Build

Preferences		— 🗆 X
type filter text	Workspace	🌣 🔻 🖒 🔻 🕇
<ul> <li>General</li> <li>Appearance</li> <li>Compare/Patch</li> <li>Content Types</li> <li>Editors</li> <li>Globalization</li> <li>Keys</li> <li>Network Connections</li> <li>Notifications</li> <li>Perspectives</li> <li>Search</li> <li>Security</li> <li>Startup and Shutdown</li> <li>Ul Responsiveness Monitoring</li> <li>Web Browser</li> <li>Workspace</li> <li>Build Order</li> <li>Linked Resources</li> </ul>	See <u>Startup and Shutdown</u> ' for workspace st. Build automatically Build automatically Befresh using native hooks or polling Saferesh on access Save automatically before build Always glose unrelated projects without pu Workspace save interval (in minutes): 5 Workspace name (shown in window title): w Workspace path: C\COBOL\COBOLIT\DevSt Show workspace path in window title Open referenced projects when a project is o Always _ Never  Prompt	artup and shutdown preferences. rompt orkspace udio200\workspace opened
> COBOL > Dynamic Languages > Help > Install/Update	Command for launching system explorer: ex	cplorer /E_/select=\${selected_resource_loc}
> Mylyn	Default (Cp1252)	Default (Windows)
<ul> <li>Remote Systems</li> <li>Run/Debug</li> </ul>	O <u>Q</u> ther: Cp1252 ⊻	Other: Windows ~
> Team > Terminal		Restore Defaults Apply
0		OK Cancel

## Window>Preferences>COBOL>License file path

The Enterprise Edition License file is required in order to use the Developer Studio with the Enterprise Edition of the Compiler, and Runtime.

The Enterprise Edition license file can be named citlicense.xml, or a variant, as long as the license name begins with "citlicense-" and has the .xml suffix. Alternatively, if you provide a non-standard name for your license, or if you choose to locate your license file in a non-default location, you should use the COBOLIT\_LICENSE environment variable to locate your license. For these cases, the COBOLIT\_LICENSE environment variable must be set in the shell in which you start the Developer Studio when you launch the Developer Studio. For details, see the Compiler and Runtime Reference Manual.

The Enterprise Edition license file is provided to Customers with Subscriptions. The default





behavior of the COBOL-IT Compiler Suite Enterprise Edition is to check for the file citlicense.xml in the default installation directory, or the directory described by the environment variable COBOLITDIR.

Use the Browse interface in Window>Preferences>COBOL to locate the license file, select it, and click on "Apply". Click "OK" to close the Window>Preferences>COBOL dialog window. If the license file is in a non-default location, or has a non-default name, you must make sure that COBOLIT\_LICENSE is set when you launch the Developer Studio, and that COBOLIT\_LICENSE includes the full path and full name of the license file.

type filter text	COBOL	φ.	\$	*
type filter text > General > COBOL > Dynamic Languages > Help > Install/Update > Mylyn > Remote Systems > Run/Debug > Team > Terminal	COBOL         General COBOL settings         License file path:         C:\Cobol\CobolIT/Citlicense.xml         Registered to:         COBOL-IT Demo         Rue de la Caserne, 45         B-1000 Bruxelles         Belgique         Subscription: Distributor - Personal use Only         Expires: 2017-12-31         Platforms: zLINUX, HPUX, AIX, WINDOWS, LINUX, SOLARIS         Products: SQL_STNTAX-ONLY, COMPILER, IDE, RUNTIME, REPORT-WRITER, EBCDIC         Created by:         COBOL-IT Demo         Rue de la Caserne, 45         B-1000 Bruxelles         Belgique	• •	Brov	* vse
	Restore Defaults	Ar	ply	

#### Renaming or placing the license file in a non-default location

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

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

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

In Windows:

>SET COBOLIT\_LICENSE=C:\COBOL\CobolIT\license\mycitlicense.xml

Select this license file from the Window>Preferences>COBOL dialog screen.





# Window>Preferences>Run/Debug>Perspectives>...

In the Panel on the left, Select Perspectives. In the option labeled "Open the associated perspective when launching", select the radio-button titled "Always". This will cause the Developer Studio to switch to the Debugger Perspective, when any Debug function is activated.

Preferences			σ	$\times$
type lifter text	Perspectives		• • •	
<ul> <li>General</li> <li>Gorond</li> <li>Daylo</li> <li>Daylo</li> <li>Denoted Lipson</li> <li>Myra</li> <li>Bernord-Systems</li> <li>Marylow</li> <li>Fairch/Rog</li> <li>Fairch/Rog</li> <li>Jann-Pring</li> <li>Salaysian</li> <li>Salaysian</li> <li>Salaysian</li> </ul>	These setting sing represents on training and debugging Appropriate can axionatically span when laveling is when an application support.         Open the association presents when laveling is a space stress of the space stress o	bult	δροι	> >
Ø	OK		Cancel	

# Window>Preferences>COBOL>Compiler (Compiler flags & Scripts)

Compiler flags can be set within the Window>Preferences>COBOL>Compiler interface. Compiler flags & scripts that are set in this interface are automatically inherited by all of the projects in the Workspace.

Compiler flags & scripts can be set at the Project level within the Project>Properties>COBOL Properties wizard.

# The File>New Wizards

# Key Concepts

The "New COBOL Project" wizard creates a sub-folder under the Workspace folder with the name of the project.

The "New Folder" wizard adds sub-folders under the Project folder. These folders can be used to organize other project elements such as copy files, list files, object files, and data files.

The "New COBOL Program" wizard opens a blank COBOL program in the COBOL-IT Code Editor, and Saves it in the selected folder.

## File>New>COBOL Project

Select File>New>COBOL Project... to open the COBOL Project Wizard. Create a new project in the Workspace.



Last playagate Search Broject	udio - COBOL-II L	Peveloper Studio		- 0 ×
New	Alt+Shift+N >	P COBOL Project	X	Quick Access
Open File		📑 Project		2 V P
Open Projects from File System Close	Ctrl+W	COBOL Program Folder	Create a COBOL project Create a COBOL project in the workspace or in an external location	vailable.
Llose All	Ctrl+Shift+W	Untitled Text File		
save Save As	Ctri+S	Ctrl+N	Project name: project1	
Save All	Ctrl+Shift+S		Contents	
Revert		-	Create new project in workspace	
Move	53		O Create project at existing location (from existing source)	
Refresh	F2		Host: Localhost	
Convert Line Delimiters To	>			
Print	Ctrl+P		Elisercity. Cricoport/coport/fobort/peasinglosco/workshace/biolecti	
Switch Workspace	>		Working sets	
Restart		-	Add project to working sets New	
Import			Working sats	
Properties	Alt+Enter		Projekting detak	
Fyit				
	🖺 Proble	ems 🛱 🖾 Console 🦷 Progress		\$ ~ =
	0 items			

Enter a project name. In this example, we create a project called project1.

In this case, entering project1 as a Project name causes a subfolder called project1 to be created under the workspace folder.

Here we are creating a new project in the workspace. Eclipse artifacts, such as the .project file, will be created in this project directory. Click [Finish].

## **Project>New>Folder**

Now that we have created a COBOL project folder, we can create subfolders for the various components in our project, such as copy files, list files, object files, and data files. To do this, we will use the New Folder Wizard.

Select project1 in the Navigator Window. Right-click on the project, and select New>Folder to enter the new folder wizard. In the new folder wizard, enter a folder name "copy" and press the [Finish] button. [Finish] closes the new folder wizard.

Repeat multiple times to create the subfolder "lst" for list files, the subfolder "object" for object files, and the subfolder "data" for data files.



		′L			THE DASIUS			V	ersion 2
workspace e Edit Nav • 🗑 🕼 🗑	- COBOL-IT Developer Studio - COBOL-IT De rigate Search Project Run Window Help	veloper Studio						Quick A	Access
Navigator 🛛					- 8	BE Outline ≅			\$ V - 0
➢ project1	New Go Into Open in New Window Copy Paste Zelste Moye	>	COBOL Project       Project       COBOL Program       Folder       File       Untitled Text File       Qther       Ctrl+N		Wew Folder Folder Create a new folder resource.	An outline is r	not avail	×	
	Rename Import Export				Enter or select the parent folder:				
	Build Project Refresh Close Project Close Unrelated Projects				Project 1				
	Show in Remote Systems view <u>R</u> un As Debug As Restore from Local History	>							
	Cobol Source Control T <u>e</u> am	>			Folder name: copy				
	Compare With Configure	>	Progress	Reso					e U
	Pamous from Contact     Ctdu	Alt - Chift - Davie							

We now have a base project directory, and 4 subfolders, in which to do our Getting Started development exercises.

#### New Folder Advanced Functions

New Folder Advanced Functions allow you to add folders to your Project which are not located as subfolders of the Project folder on the disk.

It is common that in a corporate work environment, copyfiles will be stored in a folder that all users share. This folder can be referenced by the -I [copypath] compiler flag without adding the folder to the project, so it is not strictly a requirement that it be part of the project.

However, the copyfile folder does have to be in the Project in order for the the Project to execute certain functions like, "Open Declaration", for example. Or, if your copy file contains Procedure Division code, then the copyfile folder has to be in the Project in order for the Debugger to step through the Procedure Division code statements.

So, if your copyfiles are not in a subdirectory located underneath the Project, how do you get them into the Project? That is done at the point where you create a New Folder. In File>New>Folder, ( the New Folder dialog Screen ), click on the <<Advanced button.



COBOL-IT CO	BOL-IT Developer Studio– Getting Started The Basics
Image: Second secon	Browse For Folder X Select the link target.
Folder pame:       copy         <	Coborc Copy Coby Eolder: Copy Bake New Folder OK Cancel

This expands the New Folder dialog screen to include 3 radio buttons, which are:

() Use default location

() Folder is not located in the File System (Virtual Folder)

() Link to alternate location (Linked Folder)

The default is the first one. When you are using the default setting, and you create a folder, it is created as a subdirectory of the Project. You can then create folders under this folder, and so forth...

The Virtual Folder selection allows you to add a folder on a remote machine to your project. When using the Virtual Folder, you should also select the Remote System on which the virtual folder resides from the "Choose File System" dropdown box.

The Linked Folder allows you to link to an existing folder on your system that is not located under your Project folder. For this case, you name the folder in the Folder Name entry field, but link this folder name to another location, which you browse and select. When you do this, you will find that this folder is added to your Project, and the Developer Studio and Debugger will execute functions like "Open Declaration", and "Step Into" code located in a copyfile with no problems.

## Project>New>COBOL program

We can now create a new COBOL program in the base project directory. We will create `our sample program, hello.cbl, in the project directory.

In the Navigator Window, select the project by clicking on it. This selection will ensure that the Wizard will store the project folder as the default location for the new COBOL program, which is what we want.



Version 2.0



Select File>New>COBOL Program to open the New COBOL Program Wizard. Enter the name of your new COBOL program, and press the [Finish] button. Pressing [Finish] closes the New COBOL Program Wizard.

Sile Edit N	e - COBOL-IT Developer Studio - Co avigate Search Project Run Win	OBOL-IT Developer Studio Idow Help										_	٥	$\times$
📑 🕶 🖬 🕼	B 4 ▼ 6 / ▼ 1 2 ▼ 5 ▼ 5											Quick A	ccess	2
🌤 Navigato	r≍ ⇔≎@ ∃\$\$ ₽ ⊽ ⊓ ⊟									BE Outline	8		ş	
C C C C C C C C C C C C C C C C C C C	1 New Go Into Open in New Window Copy Paste Delete Moye Rename Import Export Build Project Refresh Close Unrelated Projects	>	Image: COBOL Project         Project         Image: COBOL Program         Image: CoBOL Program		Create Pro Create new This wizard of Source folder: Name:	gram COBOL Program eates a new COBOL project1 hello.cbl	. Program.			An outline i	is not available	Ð,		
	Show in Remote Systems view Run As Debug As Restore from Local History Cobol Source Control Team Compare With	> > > > > > > > > > > > > > > > > > > >	e ≒Progress		2				Einish		Cancel		2 12	~
	Configure	>	^		0	0.11			*					
	Properties				Resource	Path		Location	Type					
3	Remove from Context	Ctrl+Alt+Shift+Down	]											
project1														

#### The Code Editor

You will see that you now have an empty COBOL program called hello.cbl in your Developer Studio Code Editor. The Outline Window becomes populated as you create hello.cbl.







#### Enter the code for hello.cbl.

🞯 workspace - COBOL-IT Developer Studio - C	COBOL-IT Developer Studio	– 0 ×
<u>File Edit Source Navigate Search Project</u>	<u>Run W</u> indow <u>H</u> elp	
📑 🕶 📓 🐚 📓 💷 🎋 🕶 🛛 🕶 隆 🛹 🖛	·[四 + 句 + や + ウ +	Quick Access
<ul> <li>Navigator ≅</li> <li>⇒ ⇒ Q</li> <li>≥ S</li> <li>⇒ </li> <li>≥ Copy</li> <li>&gt; ≥ data</li> <li>&gt; ≥ Ist</li> <li>&gt; ≥ object</li> <li>Ellerche</li> </ul>	<pre>e hello.cbl # 1 IDENTIFICATION DIVISION. 2 PROGRAM-ID. hello. 3 ENVIRONMENT DIVISION. 4 INPUT-OUTPUT SECTION. 5 FILE-CONTROL. 6 COPY "myfile.sl". 7 * 8 DATA DIVISION. 9 FILE SECTION. 10 COPY "myfile.fd".</pre>	© □ SE Outline © R <sub>2</sub>   D V = □ V ● IDENTIFICATION • hello V ● ENVIRONMENT ● INPUT-OUTPUT V ● DATA > 0 FILE V ● WORKING-STORAGE • file-status
	<pre>workING-STORAGE SECTION. 7 file-status PIC xx. 7 ws-dummy PIC x.      PROCEDURE DIVISION. 7 main. 8 OPEN OUTPUT myfile. 9 PERFORM write-file. 20 CLOSE myfile. 21 22 OPEN INPUT myfile. 23 PERFORM read-file. 24 CLOSE mufile. 24 24 25 OPEN Status PIC x 26 OPEN Tead-file. 27 OPEN Tead-file. 28 OPEN Tead-file. 29 OPEN Tead-file. 20 OPEN Tead-file. 20 OPEN Tead-file. 20 OPEN Tead-file. 21 OPEN Tead-file. 22 OPEN Tead-file. 23 OPEN Tead-file. 24 OPEN Tead-file. 25 OPEN Tead-file. 26 OPEN Tead-file. 27 OPEN Tead-file. 28 OPEN Tead-file. 29 OPEN Tead-file. 20 OPEN Tead-file. 21 OPEN Tead-file. 23 OPEN Tead-file. 24 OPEN Tead-file. 24 OPEN Tead-file. 25 OPEN Tead-file. 26 OPEN Tead-file. 27 OPEN Tead-file. 27 OPEN Tead-file. 28 OPEN Tead-file. 29 OPEN Tead-file. 20 OPEN Tead-file. 24 OPEN Tead-file. 25 OPEN Tead-file. 26 OPEN Tead-file. 27 OPEN Tead-file. 27 OPEN Tead-file. 28 OPEN Tead-file. 29 OPEN Tead-file. 20 OPEN Tead-file. 27 OPEN Tead-file. 28 OPEN Tead-file. 29 OPEN Tead-file. 29 OPEN Tead-file. 20 OP</pre>	<ul> <li>ws-dummy</li> <li>♥ ● PROCEDURE</li> <li>main</li> <li>write-file</li> <li>read-file</li> </ul>
	Problems      □ Console      □ Progress	R. 🖬 🚱 📑 🔍 🔫 🗖 🕶
	COBOL-IT Compiler	
		^
		~
	C .	

The Outline Window auto-updates as you type your COBOL program.

The Divisions (Procedure Division, Environment Division, etc...) in your program are marked with a blue D, the Sections (Working-Storage, etc...) are marked with a Green S.

The Outline Window provides a useful navigation tool. Click on a title, and navigate directly to that line of source in the COBOL program.

#### Save your COBOL program

You can SAVE your COBOL program by clicking on the Save button on the Developer Studio toolbar.

Additionally, the Window>Preferences>General>Workspace dialog screen contains interfaces for setting:

Save Automatically before build

Workspace save interval (in minutes)

Workspace save interval ensures that your work will be automatically saved at specified intervals. Save automatically before build ensures that even if you forget to save your work before starting a Build, the Developer Studio will automatically save the code changes before beginning the build.

# Project>Properties>COBOL Properties (Compiler flags & Scripts)

Compiler flags and scripts can be set at the Workspace level in the Window>Preferences>COBOL>Compiler dialog screens.

Compiler flags and scripts can be set at the Project level in the





Project>Properties>COBOL Properties dialog screens.

COBOL-IT Compiler flags are grouped by classifications that can be seen in the panel on the left.

Properties for project1				
type filter text	COBOL Properties			⇔ ▼ ⇔ ▼ ▼
> Resource	Finable source settings	Use external make		
Builders				
COBOL Formatter				
<ul> <li>COBOL Properties</li> </ul>				
Accept/Display				
CALL Parameters				
Cmd Flags				
Customize Setup				
> Data Handling				
> DevOps Tools				
> Dialects				
Environment				
> File Handling				
Informational				
> Link				
Optimize Options				
Pre-processing Options				
Project Options				
Remote Source Control				
Standard Options				
Warnings				
Project References				
Refactoring History				
Run/Debug Settings				
> Task Repository				
Task Tags				
WikiText				
			Restore Defaults	Apply
(?)		l	OK	Cancel

#### Select the Enable source settings checkbox to enable compiler flag and script settings.

When you have selected the Enable source settings checkbox, it will be enabled at the top of all of the compiler flag screens.

The compiler flag screens are listed in the Navigator Window.

#### Accept/Display

Affect the behavior of the ACCEPT and DISPLAY statements.

Accept/Display	¢ 🔻	<> ▼	•
✓ Enable source settings	Use external make		
ACCEPT assigns WITH AUTO attribute by a ACCEPT assigns WITH UPDATE attribute b DISPLAY PIC 9 field (no sign, no decimal)	default (-faccept-with-auto) y default (-faccept-with-update) as it is in memory (-fraw-pic9-display)		

#### -faccept-with-auto

Causes the WITH AUTO clause to be assumed by default on a field-level ACCEPT statement.

#### -faccept-with-update

Causes field-level ACCEPT statements to assume the "WITH UPDATE" clause.

-fraw-pic9-display (Internal use only)





#### DISPLAY PIC 9(X) (no sign, no decimal) as it is in memory.

#### CALL Parameters

#### Affect the behavior of the CALL statement.

CALL Parameters	⟨→ ▼ ⇒ ▼				
✓ Enable source settings	Use external make				
Use COMP-5 when passing COMP in CALL	.parameter (-fcall-comp5-as-comp)				
Pass NULL terminating pointer on CALL sta	itement (-fnull-param)				
Output static C function calls for the CALL statement (-fstatic-call)					
CALL BY VALUE assigns SIZE IS AUTO by d	efault. (-fvalue-size-is-auto)				
Conversion					
• Cause CALL BY VALUE of PIC X to not be	converted to numeric COMP-5 (-fraw-by-value)				
$\bigcirc$ CALL BY VALUE of PIC X is converted to n	umeric COMP-5 (-fno-raw-by-value)				

#### -fcall-comp5-as-comp

On little-endian platform (intel Linux, Windows) when a call USING clause contains a literal, causes the literal to be copied as a COMPUTATIONAL value, rather than as a COMP-5 value.

#### -fnull-param

Causes an extra NULL pointers to be passed as the last argument on CALL statements.

#### -fstatic-call

Causes static C function calls to be generated for the CALL statement.

#### -fvalue-size-is-auto

Causes the SIZE IS clause in a CALL USING BY VALUE statement to be set to AUTO.

#### -fraw-by-value

Causes PIC X to not be converted to numeric COMP-5 in a CALL USING BY VALUE statement. This is the default.

#### -fno-raw-by-value

Causes PIC X to be converted to numeric COMP-5 in a CALL USING BY VALUE statement.

#### Cmd Flags

Allows for the setting of additional user flags. Enter additional compiler flags in the entry field and they will be added to your compile string.

Cmd Flags		\$ ▼ \$ ▼ ▼
✓ Enable source settings	Use external make	
Additional user flags		

#### **Customize Setup**

Allows for the scheduling of scripts to be run before/after clean and build operations.





Customize Setup			↓ ↓ ↓ ▼
Enable source setting	5	Use external make	
Customize Setup			
Build/Clean Setup			
Build command		citmake	<u>B</u> rowse
Pre-build script		%COBOLITDIR%\setenv_cobolit.bat	<u>B</u> rowse
Post-build script			<u>B</u> rowse
Pre-clean script		%COBOLITDIR%\setenv_cobolit.bat	<u>B</u> rowse
Post-clean script		%COBOLITDIR%\setenv_cobolit.bat	<u>B</u> rowse
Compiler command		cobc %f	<u>B</u> rowse
Compiler execution dire See "Compiler comman for code detail	ectory nd"	%p	<u>B</u> rowse
Run/Debug Setup			
Pre-run script	%	COBOLITDIR%\setenv_cobolit.bat	<u>B</u> rowse
Post-run script			<u>B</u> rowse
Pre-debug script	debug script %COBOLITDIR%\setenv_cobolit.bat		<u>B</u> rowse
Post-debug script			<u>B</u> rowse
Remote SSH port	Ren	note SSH port range	
8484	50		
%p Module absolute pat %P Project Absolute patl %m Module name (no p	h n <sup>d</sup> ath no	%r Module relative path %f Module file name (no path) • extension)	

Browse and select scripts which are automatically executed in certain conditions :

Pre-build script	Prior to a proj	Prior to a project build.				
Post-build script	After the com	After the completion of a project build				
Pre-clean script	Prior to the ex	Prior to the execution of the clean operation				
Post-clean script	After the com	pletion of the clean operation				
Compiler command	cobc %f	%f = Module file name (no path)				
Compiler execution d	irectory %p	% p = Module absolute path				
Pre-run script	Prior to launch	h of a run of the application				
Post-run script	After the completion of the run of the application					
Pre-debug script	Prior to launch	h of the application in the debugger				
Post-debug script	After the com	pletion of the run of the application in the debugger				
Remote SSH Port	8484					
Remote SSH port ran	ge 50					

### Data Handling>Data Operations

Affects the behavior of data operations.





Data operations	← ← ⇒ → →
☑ Enable source settings	
Align 01/77 data items on 8-byte boundary (-falign-8)	
Make compare of numeric with PIC X using numeric value of PIC X (-fnum	eric-compare)
Slide location of fields in memory after OCCURS DEPENDING ON table (-fo	do-slide)
Typedef global scope	
• Cause TYPEDEF to have a GLOBAL scope (-fglobal-typedef)	
O Does not cause TYPEDEF to have a GLOBAL scope (-fno-global-typedef)	
Numeric display sign	
Numeric display sign ASCII (-fsign-ascii)	
O Numeric display sign EBCDIC (-fsign-ebcdic)	
Signed numerics assume	
By default signed numerics assume SIGN IS TRAILING	
O Signed numerics assume SIGN IS LEADING by default (-fsign-leading)	
$\bigcirc$ Signed numerics assume SIGN IS SEPARATE by default (-fsign-separate)	
Round COMP-1/COMP-2 on MOVE to non COMP-1/COMP-2 (-fround-fp)	
UNSTRING uses MOVE PIC X to the INTO field instead of a raw copy (-funs-	tring-use-move)
Recording Mode	
RECORDING MODE not assigned if unspecified	
O RECORDING MODE assigns V if unspecified as default (-freemode-v)	
KELOKDING MODE assigns F as default always (-freemode-f)	
RECORD DEPENDING clause is compatible with ISO (-frecord-depending-is	0)
□ Initialize FD at program entry (-finitialize-fd)	

#### -falign-8

Aligns 01-level and 77-level data on 8 byte boundaries.

#### -fnumeric-compare

Causes the comparison of a numeric field with a PIC X field to interpret the value of the PIC X field using its numeric value.

#### -fodo-slide

Causes data items that appear after a variable-length table (with OCCURS DEPENDING clause) to always immediately follow the table, whatever the current size of the table. The internal addresses of these data items change as the table's size changes.

#### -fround-fp

Affects behaviors when COMP-1 or COMP-2 are "moved" into non-COMP-1 or COMP-2 target fields when the target field has fewer decimal places than the source field. Causes the value to be rounded to the number of decimal of the target field.

#### -funstring-use-move

When an UNSTRING INTO operation is described as PIC 9, causes the operation UNSTRING operation to be performed using a MOVE operation instead of raw copy operation. Rules defined by the move-picx-to-pic9 compiler configuration flag are used for conversion.

#### -fglobal-typedef





Causes TYPEDEFs to be GLOBAL for all nested programs.

#### -fno-global-typedef

Causes TYPEDEFs to be local to the current program.

#### -fsign-ascii

Causes numeric DISPLAY items that include signs to be interpreted according to the ASCII sign convention. (default on ASCII machines)

#### -fsign-ebcdic

Causes numeric DISPLAY items that include signs to be interpreted according to the EBCDIC sign convention. (default on EBCDIC machines)

#### -fsign-leading

Makes SIGN IS LEADING the default.

#### -fsign-separate

Makes SIGN IS SEPARATE the default.

#### -frecmode-v

Causes all unspecified RECORDING MODE clauses to be interpreted as RECORDING MODE V.

#### -frecmode-f

Causes all unspecified RECORDING MODE clauses to be interpreted as RECORDING MODE F.

#### -frecord-depending-iso

For ISO-compatibility, causes files declared with a RECORD DEPENDING ON <FIELD> clause, without any FROM or TO value, to assume a FROM and TO value of the maximum record size.

#### -finitialize-fd

Causes records declared in the File Section to be initialized when program is loaded in memory.

#### Data Handling>Linkage Section

Affects the handling of data in the Linkage Section.





Linkage Section	♦ ♥ ♀ ▼
Enable source settings	Use external make
Allocate unused LINKAGE item ( -falloc-un	used-linkage ) ors ( -forotost-linkago )
Avoid SIGVEC in debugger when paramet	ers are not provided ( -fsafe-linkage )

#### -falloc-unused-linkage

Causes the compiler to allocate static memory for level 01 fields in the Linkage Section that are not used in either a USING clause or an ENTRY clause.

#### -fprotect-linkage

Generates code at the entry point of a program containing a USING xxx clause.

#### -fsafe-linkage

Generates code at the entry point of a program containing a USING xxx clause. This allows for the omission of parameters. Doing this will avoid a SIGVEC being returned by the debugger when all linkage parameters are not provided.

#### DevOps Tools>Checkpoint/Restore

Allows for use of the Checkpoint/Restore functionality.

Checkpoint/Restore		⇔ ▼ ⇔ ▼
Enable source settings	Use external make	
Enable checkpoint by saving context for sa	ve/reload (-fcheckpoint)	

#### -fcheckpoint

Enables setting of checkpoints. Program state is saved at checkpoints, and can be reloaded.

#### DevOps Tools>Code Coverage

Allows for the configuration of Code Coverage functionality.

Code Coverage		← ▼ ⇔ ▼ ▼
Enable source settings	Use external make	
✓ Open coverage view automatically □ Apply Code Coverage to runtime session ( -code-cover )		
Store code coverage information ( -debug	db ) debugdb.dbd	
See <u>Label Decorators</u> for coverage decorators.		
See Annotations for editor highlighting style.		

#### **Open coverage view automatically**

Causes the Coverage View to open automatically when compiling with -code-cover -code-cover -debugdb=debugdb.dbd





Causes code coverage information to be stored in the debugdb.dbd file. For more information on Code Coverage see Getting Started using the Developer Studio- The Utilities.

#### DevOps Tools>Data Displayer

Allows for the configuration of Data Displayer functionality

Data Displa	ayer		⇔ ◄ ⇔ ▼ ▼
✓ Enable so	urce settings	Use external make	
Load data	partially		
Page size:	0		
Use setting	gs from a XDD file		
Use var	iable size record format		
Use MF	Line Sequential Escape format		
Generate o	c-TreeACE .xdd file ( -fgen-xdd )		

#### Load data partially

For use with table-mode display. Sets a page size which is number of records that will be written to the table. In large files, tables can be browsed by page, with the table re-loading by page, instead of loading all of the records at once.

#### Use settings from a XDD file

Data Displayer reads information from the XDD file.

#### Use variable-size record format

Data Displayer evaluates variable-size relative file headers.

#### **Use MF Sequential Escape format**

Corresponds to compiler configuration file setting lin-seq-mf: yes. Supports line sequential files that preface bytes with values less than 0x20 with 0x00.

#### -fgen-xdd

Causes the compiler to parse FDs in the program and generate .xdd files.





### DevOps Tools>Debugging Tools

Enables a range of debugging functionalities.



#### -debug

Enables all run-time error checking.

#### -ftrap-unhandled-exception

Enhances information dump when program aborts and ON EXCEPTION/ON OVERFLOW language is not present in the COBOL program.

#### -g

Produce debugging information in the output.

#### -fdebugdb

When used with –g, store alls debugging information into a file name <modulename>.dbd.

#### -fsource-location

Generates source location code, enabling information to be dumped on source location when the runtime aborts. Enabled by the –g compiler flag, and by the –debug compiler flag.

#### -fstack-check

Enables stack checking debug function. Enabled by the –g compiler flag, and by the –debug compiler flag.

#### -G





Produces debugging information in the output, allowing "C"-level debugging. Use with -fnostrip. **-fdebug-exec** 

Enables the debugging of code generated by a pre-processor when using the -preprocess compiler flag.

#### -fdebugging-line

Enables support for debugging lines. (Source lines that contain 'D' in indicator column)

#### -fmem-info

Enables Dump of Working-Storage when runtime aborts. Use with the -g or -debug compiler flag.

#### -fnostrip

Causes objects and object and executable files to NOT be stripped. Stripping an object or an executable is the action of removing system level debugging information.

#### -ftraceall

Generates trace output at runtime, listing SECTION/PARAGRAPH/STATEMENTS names as they are executed.

#### -ftrace

Generates trace output at runtime, listing SECTION/PARAGRAPH/STATEMENTS names as they are executed.

#### -fsimple-trace

Generates trace output at runtime for executed SECTION/PARAGRAPHs.

#### -fsplit-debug-mark

DEBUG marks respect max 72 characters (default).

#### Debugger socket connection timeout in seconds

Debugger socket connection timeout, set to 10 seconds by default.





### DevOps Tools>Profiler

Allows configuration of the Profiler function.

Profiler		↓ ↓ ↓ ▼ ↓
Enable source settings	Use external make	
Open Profiler view automatically		
Generate paragraph profiling code ( -	-fprofiling )	
Enable interval profiling		
Refresh interval (seconds) 5		

#### **Open Profile view automatically**

Causes the Profiler View to open automatically when compiling with -fprofiling

#### -fprofiling

Causes profiler information to be stored in a .xls file. For more information on Profiling, see Getting Started using the Developer Studio- The Utilities.

#### Enable interval profiling

Allows interval profiling, which is displayed in the runtime tab of the profiling view.

#### **Refresh interval (seconds)**

For the runtime tab of the profiling view, affects how often the memory/cpu usage graphs are refreshed.

#### **Dialects>Compiler Configuration**

Allows selection of compiler configuration file.



#### -conf=<file>

Causes a user-defined compiler configuration file to be referenced either as the default configuration file.





#### -std=<dialect>

Causes one of the dialect-oriented compiler configuration files to be used instead of the default compiler configuration file.

#### Dialects>IBM Compatibility

Provides refined compatibility with IBM COBOL.

IBM Compatibility	♦ ♥	
Enable source settings	Use external make	
Generate CICS-compatible code ( -fcics )		
COMPUTE statement uses IBM rules for number of decir	mal places used storing intermediate results ( -fcompute-ibm )	
Cause DISPLAY statement to behave like IBM Mainframe	e COBOL DISPLAY ( -fdisplay-ibm )	
Compile for a EBCDIC charset runtime ( -febcdic-charset	•)	
Enable IBM listing extensions (TITLE, SKIP 1/2/3, EJECT) (	-fibm-listing-macro )	
Set compiler and runtime in IBM Mainframe compatibility mode ( -fibm-mainframe )		
If first sub-field is SYNC then group is SYNC ( -fibm-sync )		
Causes variable-length blocked files to have READ/WRI	TE behaviors compatible with the Mainframe ( -fmainframe-vb )	
Cause executing program to always use region 0 ( -freg	ion0 )	

#### -fcics

Generates CICS-compliant code.

#### -fcompute-ibm

Causes arithmetic expressions (like a+B\*c) in COMPUTE statements, and comparisons to use IBM COBOL defined rules for determining the number of decimals used in intermediate results.

#### -fdisplay-ibm

Causes the output of the DISPLAY Statement for numeric fields to be more compatible with IBM mainframe.

#### -fibm-listing-macro

Enables IBM listing extensions (TITLE, SKIP1/2/3, EJECT ...) (default)

#### -fibm-mainframe

Causes the compiler and runtime to operate in an IBM Mainframe compatible mode.

#### -fibm-sync

Applies SYNC attribute to group item if first elementary field is described with the SYNC attribute. (default).





#### -fmainframe-vb

Causes WRITEs and READs of Variable Blocked files to assume formats compatible with the Mainframe Z/OS COBOL Format.

#### -fregion0

Causes the module to always switch to region 0 even if called from another region.

#### Dialects>Micro Focus Compatibility

Provides refined compatibility with Micro Focus COBOL.

Micro Focus Compatibility	♦ ▼ \$\circ\$	<b>•</b> •
✓ Enable source settings	Use external make	
Allow use of intrinsic functions without the FUNCTION ke	yword ( -ffunctions-all )	
Treat lies with '*' or '/' in column 1 as comments ( -fmfcor	mment )	
Cause syntax parser to be more compatible with Micro Fo	ocus ( -fmf-compat-parser )	
Cause syntax parser to be MF compatible with control ch	aracters escaped by 0 (-fmf-ctrl-escaped-parser )	
Cause runtime to return file status 05 for optional files ( -	fmf-file-optional )	
Emulate MF directive HOSTNUMCOMPARE ( -fmf-hostnu	mcompare )	
Create relative files compatible with MF format ( -fmf-rela	ativefile )	
Do not truncate intermediate fields according to PIC in ca	lcuations with binary fields ( -fnotrunc )	
Cause divide-by-zero to generate an exception ( -fdiv-che	eck)	

#### -ffunctions-all

Allows use of intrinsic functions without the FUNCTION keyword

#### -fmfcomment

Treats lines with '\*' or '/' in column 1 as comments.

#### -fmf-compat-parser

Increases compatibility of syntax parser with the Micro Focus syntax parser (default)

#### -fmf-ctrl-escaped-parser

Syntax parser is MF compatible with control character escaped by 0 (default).

#### -fmf-file-optional

Causes files declared as OPTIONAL and OPEN in EXTEND to return file-status code "05" if the file is created and file-status code "00" if the file exists.

#### -fmf-hostnumcompare

The -fmf-hostnumcompare compiler flag affects comparisons of USAGE DISPLAY numeric data items when one of the numeric data items in the comparison contain non-numeric data.

#### -fmf-relativefile

The –fmf-relativefile compiler flag causes the runtime to assume the Micro Focus format for relative files for both READ and WRITE operations.

#### -fnotrunc





Causes truncation of binary fields to NOT be made according to the PICTURE clause while performing intermediate computations.

#### -fdiv-check

Causes divide-by-0 operations to generate an exception.

#### Dialects>Other COBOL Compatibilities

Provides refined compatibility with other COBOL compilers.

Other COBOL Compatibility		⇔ ▼ ⇔ ▼ ▼
✓ Enable source settings	Use external make	
LIKE clause behavior compatible with AS400 ( -fas400-like )		
Use CA REALIA sign coding for USAGE DISPLAY ( -fcarealia-sign )		
Emulate GCOS operation ( -fgcos-mode )		

#### -fas400-like

Causes the LIKE clause to act compatibly with the AS400 implementation. A field declared with the LIKE clause is described as a PIC X (other field's byte size).

#### -fcarealia-sign

Use CA Realia sign coding for Usage Display.

#### -fgcos-mode

Causes the compiler to more closely emulate GCOS operations.

#### Environment>Copy Files

Affects the handling of COPY files. In our sample, we use a copy folder outside the Project folder.

Copy Files		⇔ ▼ ⇔ ▼ ▼
✓ Enable source settings	Use external make	
Add default copy/include file extensions, separated by ";" (-	ext <extension>)</extension>	
Add <directory> to copy/include search path ( -I <directory< td=""><td>/&gt; )</td><td></td></directory<></directory>	/> )	
		Ne <u>w</u>
		<u>R</u> emove
		<u>U</u> p
		Dow <u>n</u>
		Browse
Add mark for beginning and and of COPY statement in	listing and pre-processed file ( -fcopy-mark )	
Search for COPY		
• Search for COPY file first in current directory ( -fcurdir-in	nclude )	
$\bigcirc$ Do not search for COPY file first in current directory ( -f	no-curdir-include )	
Translate COPY file to		
Iower case ( -ffold-copy-lower )		
O upper case ( -ffold-copy-upper )		
Keep copy statements in listing and pre-processed files	(-fkeep-copy-statement)	

#### -ext <extension>

Adds <extension> to list of default copy file extensions. For example, to direct the compiler to search for copy files with .fd and .sl extensions, use the compiler flags "-ext fd -ext sl".





#### -I <directory>

Allows location of copy files in <directory>.

#### -fcopy-mark

Adds mark for begin/end of COPY In listing and preprocessed file.

#### -fcurdir-include

Causes COPY files to first be searched for in the current directory, before locations described with the -I <Path>, or with environment variables.

#### -fno-curdir-include

Causes COPY files to not be searched for in the current directory, before locations described with the -I <Path>, or with environment variables.

#### -ffold-copy-lower

Folds COPY file names to lower case.

#### -ffold-copy-upper

Folds COPY file names to upper case.

#### -fkeep-copy-statement

In listing and preprocessed file, keep COPY statements.

### Environment>Encoding

Allows UTF-8 and UTF-16-LE encoding.

Encoding		↓ ↓ ↓ ↓ ↓
Enable source settings	Use external make	
PIC N uses UTF-16-E encoding (default is UTF-16-BE) ( -futf16-le )		
Source files use UTF-8 encoding ( -futf-8 )		

#### -futf16-le

Causes fields declared as PIC N to be stored as UTF16-LE (Little Endian). Note that by default, fields declared as PIC N are stored as UTF16-BE (Big Endian).

#### -fut-8




Instructs the compiler that the source file, and literals are UTF-8 encoded. The -futf-8 compiler flag can be used with, or without the -codepage compiler flag.

## Environment>Source Files

Affects how source is interpreted.

Source Files	⇔ ▼ ⇔ ▼ ▼
✓ Enable source settings	Use external make
Source format O Assume free (Terminal) source format ( -free ) O Assume fixed (ANSI) source format ( -fixed )	
Set codepage to cp_id (-codepage <cpid>)  Define the codepage of the source file and literals (-sour List file created with -t compiler flag can be compiled (-fl Enable hyphen placed before the beginning of a continu</cpid>	ce-codepage <cpid>) isting-sources) ation line (-fcontinuation-line)</cpid>

#### -free

Instructs the compiler that source code is in the free, or terminal source format.

#### -fixed

Instructs the compiler that source code is in the fixed source format. The –fixed compiler flag is assumed, by default.

#### -codepage <cpid>

Defines the encoding of PIC X in memory. If –source-codepage is specified, the compiler converts from the codepage-id used in the –source-codepage compiler flag to the codepage-id used in the –codepage compiler flag.

#### -source-codepage <cpid>

Defines the code page to be used when editing the source and the code page used for string literals in the COBOL source code.

#### -flisting-sources

Informs the compiler that source is the result of program listing option (-t <file> ).

#### -fcontinuation-line

Allows a hyphen in column 7, with no following text, to be recognized as not being a continuation line.

#### File Handling>External Files

Affects the handling of External files.





External Files	\$ ▼ \$ ▼ \$
✓ Enable source settings	Use external make
External files When a file is EXTERNAL, causes the DYNAM When a file is EXTERNAL, causes the DYNAM	IIC ASSIGN field to be implicitly EXTERNAL ( -ffile-auto-external ) IIC ASSIGN field to not be implicitly EXTERNAL ( -fno-file-auto-external )

#### -ffile-auto-external

The –ffile-auto-external compiler flag affects the way that the compiler treats variables describing file-names for files described as EXTERNAL. (Default)

#### -fno-file-auto-external

Disables -ffile-auto-external. When disabling this functionality, if you have separate programs sharing the same EXTERNAL file that also have file-var fields, then changes made between the programs will not automatically be shared.

## File Handling>File names

Affects the interpretation of file names.

File names	⇔ ▼ ⇔ ▼ ▼
Enable source settings	Use external make
Disable filename conversion ( -fno-realpath )	

## -fno-realpath

Causes file names to NOT be extended to a fully qualified path.

## File Handling>Line Sequential

Affects the handling of line sequential files.



#### -fls-expand-tab

Causes the READ of a LINE SEQUENTIAL file to expand the TAB character to 8 spaces (default)

#### -fprinter-crlf

Files declared with ASSIGN TO PRINTER file names are generated with compatibility for DOS printers. This will change the End Of Record to CR/LF (instead of LF)

#### -fsequential-line

Causes all non-qualified SEQUENTIAL files to be declared as LINE SEQUENTIAL.





# File Handling>Relative Files

Affects the handling of relative files.

Relative Files		⇔ ▼ ⇔ ▼
Enable source settings	Use external make	
Cause the record header of relative files to use bigendia	n values ( -frelativefile-bigendian )	

## -frelativefile-bigendian

Causes the record header of relative files to be stored in BigEndian format.

# File Handling>Select Behaviors

Sets default behaviors for Select statements.



## -fautolock

Sets default for SELECT to LOCK MODE IS AUTOMATIC

#### -fexclusivelock

Causes all files with no LOCK MODE clause in their SELECT statement to be declared implicitly as LOCK MODE is EXCLUSIVE unless a SHARING clause in the SELECT statement or in the OPEN statement indicates otherwise.

#### -fmainuallock

Causes all files with no LOCK MODE clause in their SELECT statement to be declared implicitly as LOCK MODE is MANUAL unless a SHARING clause in the SELECT statement or in the OPEN statement indicates otherwise.

#### -fshare-all-default

The -fshare-all-default compiler flag causes all files to be declared implicitly as SHARE WITH ALL.

#### -fshare-all-manulock

Causes all files with a SHARE WITH ALL clause in their SELECT statement to be declared implicitly as LOCK MODE IS MANUAL.





#### -fshare-all-autolock

Causes all files with a SHARE WITH ALL clause in their SELECT statement to be declared implicitly as LOCK MODE IS AUTOMATIC.

#### -use-extfh <NAME>

Names an EXTFH File handler to be used, enabling the use of an external file system.

#### -ffcdreg

Allows a user of an EXTFH compliant data source to directly read and write the File Control Description (FCD) through which information passes to and from an EXTFH-compliant data source. Use with -use-extfh.

## File Handling>Sort Files

Affects the behavior of the Sort statement.

Sort Files		⇔ ▼ ⇔ ▼ •
☑ Enable source settings	Use external make	
Use EXTSM sort module ( -use-extsm <name> )</name>		
<name> of custom sort module is optional</name>		<u>B</u> rowse

#### -use-extsm <NAME>

Names a runtime module to be used, enabling the use of an external sort handler.

## File Handling>Third-party support

Enables the use of EXTFH-compliant file systems.



#### -fdbd

Activates the usage of Oracle Berkeley DB isam files.

#### -fvbisam

Activates the usage of VBISAM isam files.

#### -fdisam





Activates the usage of DISAM isam files.

#### -fctree

Activates the usage of CTREE isam files.

## -fctree-field-numbering

Causes the CTREE XDD generator to generate a prefix F <field-number> before field names. Use with –fgen-xdd compiler flag.

#### -fctree-no-full-qualification

Causes the CTREE XDD generator to not generate the fully qualified data names in the XDD description of the file. Use with –fgen-xdd compiler flag.

## Informational

Returns information, does not compile source files.



# --help

Lists compiler flags.

## --check-codepage <cpid>

Checks if a given codepage is recognized by the ICU library.

#### --list-codepage

Lists all supported codepages in the following format: Codepage: [list of synonyms for codepage]

#### --list-intrinsics

Lists supported intrinsic functions.

#### --list-mnemonics

Lists supported mnemonic names.





Version 2.0

#### --list-reserved

Lists all reserved words.

#### --version, -V

Displays compiler version.

# Link>C Code Generation

Affects the way COBOL is translated into "C".

C Code Generation	🧔 🔻 🖒 💌 👻
✓ Enable source settings	Use external make
Command line parameters are read ala C interface ( -fC-c	md-line )
Initialize C data structures in compiled object ( -fC-data-ir	nit )
nitializes the COBOL runtime system at runtime startup (	-fimplicit-init )
Generate code for gcc compiler ( -fgcc )	
On very large sources, avoid gcc infinite compilation time	bug ( -fgcc-bug )
Generate gcc goto code. Enabled with -fgcc or when CO	B_CC=gcc. (-fgcc-goto)
• Generate main() symbol when used with -x (-fmain)	
$\bigcirc$ Do not generate main() symbol when used with -x ( -fnc	-main )
Include main symbol in module object when compiled wi	ith C (-finclude-main)
Causes main symbol to be built as an object not in the lib	orary (unix only) (-fmain-as-object)

## -fC-cmd-line

When used with -x, causes the program to receive command line parameters as though they were given in C.

## -fC-data-init (Internal use only)

Controls if the C data structure created by the compiler is initialized in the source (at compilation time) or at runtime. This should not be changed.

#### -fimplicit-init

Initializes the COBOL runtime system at runtime start-up.

#### -fgcc

Generates gcc-compliant C code. The –fgcc compiler flag is enabled when COB\_CC=gcc.

Default on Linux platforms.

#### -fgcc-bug

When using a gcc compiler on very large source files, the gcc compiler could enter an infinite loop. This bug is avoided by using the –gcc-bug compiler flag.





#### -fgcc-goto

Generates gcc-computed goto code. Enabled when using the –fgcc compiler flag, or when COB\_CC=gcc.

#### -fmain

Generates main() symbol when used with -x (default)

#### -fnomain

Does not generate main() symbol when used with -x

#### -finclude-main

Causes main symbol to be included in module object when compiled with -c.

## -fmain-as-object

Generates main() symbol as object not in library (unix only) (default)

## Link>C Compiler/Linker

Allows passing options to the C Compiler or Linker

C Compiler/Linker		⇔ ▼ ⇔ ▼ ▼
✓ Enable source settings	Use external make	
Pass <define> to C compiler ( -D <define> )</define></define>		<u>B</u> rowse
Pass CC_opt to C compiler ( -Wc CC_opt )		
Pass LD_opt to C compiler ( -WI LD_opt)		

## -D <define>

Passes <define> to the "C" Compiler

#### -Wc CC\_opt

Passes CC\_opt directly to the C Compiler, where CC\_opt is a compiler flag, or string, that can be processed by the C compiler.

#### -Wl LD\_opt

Passes LD\_opt directly to the Linker, where LD\_opt is an option, or string that can be processed by the Linker.





# Link>Full Build

Affects the building of compiled objects generated by the Compiler.



#### -b

Links multiple input files into a single dynamically loadable module.

-m

Builds a dynamically loadable module. (Default).

#### -X

Builds an executable program.

#### -fmodule-uppercase

Causes the output file name to be created in upper-case, when used with the -m compiler flag.

#### -flink-only

Causes the main() symbol to not be generated, when used with -x.

#### -l <lib>

Causes the library <mylib> to be used by the linker.

#### -L <dir>

Adds <directory> to the library search path.





# -R <dir>

Adds <directory> to runtime library search path (if supported).

# -MF <file>

Writes dependency list into <file>.

## -MT <target>

Names the target file used for the dependency list.

## -fobj-cit

Causes compiled object to be generated with a cit extension instead of .dll (windows) or .so (unix/linux).

## -fmf-gnt

Causes compiled object to be generated with a gnt extension instead of .dll (windows) or .so (unix/linux).

#### -fmf-int

Causes compiled object to be generated with an int extension instead of .dll (windows) or .so (unix/linux).

# Link>Partial Build

Interrupts compilation prior to the creation of a compiled object.



-c

Compile and assemble, but do not link.

## -C

Interrupts the compilation after converting COBOL to C.

## -E

Interrupts the compilation after the preprocessing of the COBOL code, without doing any





translation to "C", compilation, assembly, or linking.

#### -S

Interrupts the compilation after after output of the assembly file. Translated C files are compiled by cc. The output is saved in a file with a .s extension.

## -fvalidate-only

Compile source, no output produced, EXEC are ignored

#### -fsyntax-only

Performs syntax error checking only. Output is limited to results of syntax check.

#### -fexec-check

Used with -fsyntax-only, checks the EXEC SQL/CICS/DLI syntax.

# **Optimize Options**

Enables optimizations of a range of functions.



# -0

Enables optimization. `-O', `-Os' and `-O2' are passed to the "C" compiler as is and used for "C"-level optimization.

#### -f77-opt

Optimizes the use of integers stored in USAGE DISPLAY or PACKED fields in level-77 data items.

#### -fall-external-link

Causes the targets of the CALL statement to all be assumed to be external-links.

#### -fbin-opt





Enables the use of CPU integers when manipulating USAGE COMP and USAGE COMP-5 data elements.

#### -fcall-opt

Enables CALL statement optimization. Programs containing CANCEL statements should not be compiled with -fcall-opt.

#### -fdecimal-optimize

Optimizes the conversion from DISPLAY/COMP-3 to binary values in COMPUTE statements.

#### -ffast-op

Enables the runtime to use faster operations when manipulating data items declared as USAGE DISPLAY or USAGE COMP-3.

#### -findex-optimize

Improves performance where indexes in tables are evaluated and USAGE DISPLAY fields are used as indexes.

#### -finitialize-opt

Optimizes the implementation of the initial field initialization at runtime startup and the execution of the INITIALIZE statement by grouping field initializations wherever possible.

#### -freturn-opt

Generates optimized PERFORM return code. The –freturn-opt compiler flag is ignored when using the –fgcc compiler flag.

#### -fcmp-opt

Activates optimizations when comparing literals with variables. (default)

#### -fno-cmp-opt

Disables optimizations when comparing literals with variables.

#### -ffast-figurative-move

Enables fast MOVEs of figurative constants. (default)

#### -fno-fast-figurative-move

Disables fast MOVEs of figurative constants.





# **Pre-processing Options**

#### Affects pre-processing functionalities.

Pre-processing Options	⇔ ▼ ⇔ ▼ ▼
✓ Enable source settings	
COPY REPLACING ==xxx== applies LEADING by default ( -fcopy-default-leading )	
COPY REPLACING ==xxx== also processes EXEC/END-EXEC block ( -fcopy-exec-replace )	
COPY REPLACING ==xxx== applies partial replacements like Micro Focus ( -fcopy-partial-replace )	
Pre-processor command (-preprocess= <cmd>)</cmd>	Browse
Expand COPY statements inside EXEC/END-EXEC blocks ( -fexpand-exec-copy )	
Expand INCLUDE statements inside EXEC/END-EXEC blocks ( -fexpand-sql-include )	
When using pre-compiler, display error at original source line ( -fkeep-org-src-line )	
Causes the makesyn compiler flag to change the output of a pre-processed file ( -fmakesyn-patch-preprocess )	
Cause REPLACE statements to nest until REPLACE OFF ( -freplace-additive )	
Cause SPCRW2 to be run after the -preprocess script ( -frw-after-preprocess )	
Define word-level synonym ( -makesyn "key=value" ), key= , value=	
SPCRW2 is run automatically if needed ( -fauto-sprwr)	
○ SPCRW2 is not run ( -fno-auto-sprwr )	

#### -fcopy-default-leading

When using the ==xxx== notation in a COPY REPLACING statement, causes the LEADING phrase to be assumed by default.

#### -fcopy-exec-replace

When a COPY REPLACING == xxx == statement is performed, causes text inside EXEC / END-EXEC blocks to be also replaced if applicable.

#### -fcopy-partial-replace

When a pattern like COPY FIC1 REPLACING == WJXX- == BY == WJ03- == is processed : If this flag is on, the preprocessor uses a partial replacement as defined by MF and ANSI2002 standard. If it is off (the default) the IBM mainframe and ANSI85 standard is used.

#### -preprocess=<CMD>

Causes <CMD> to be run after the COBOL pre-processing step. <CMD> is a script of batch file in which an external pre-processor is run.

#### -fexpand-exec-cpy

The –fexpand-exec-copy compiler flag causes the compiler to expand COBOL COPY statements inside EXEC ... END-EXEC blocks. This applies to both EXEC SQL and EXEC CICS blocks.

#### -fexpand-sql-include

Used with -E, expands 'EXEC SQL INCLUDE <File name> END-EXEC' in the -E output.

-fkeep-org-src-line





For use with the integrated pre-processor ( -preprocess ). Causes errors to be reported on the original source line.

#### -fmakesyn-patch-preprocess

Causes the makesyn compiler flag to change the output of a pre-processed file.

#### -freplace-additive

Allows for the use of the REPLACE ADD verb, which has the effect of nesting a REPLACE statement inside an existing REPLACE statement.

#### -frw-after-preprocess

Causes SPCRW2 to be run after the -preprocess script.

#### -makesyn "key=value"

Provides a way to make a reserved word a synonym for another reserved word. A common usage is to make COMP a synonym of COMP-5. >cobc –makesyn comp=comp-5 hello.cbl

#### -fauto-sprwr

Causes SPCRW2 to run automatically when needed before any -pre-process script (default).

#### -fno-auto-sprwr

Causes SPCRW2 to not run automatically when needed before any -pre-process script.

# **Project Options**

Affects handling of the build process

Project Options			<b>⇔</b> ▼ ⇒ ▼ •
✓ Enable source settings	Use ext	ternal make	
Always rebuild makefile			
Cobol program extension	o compile. Enter all extensions separated by ;		
cob;cbl;COB;CBL			
Cobol program extension	o preprocess before compile. Enter all extensions se	eparated by ;	
pco;PCO			
Setup Script	%COBOLITDIR%\setenv_cobolit.bat		<u>B</u> rowse
Working directory			Browse
(default: project directory)			

#### Always rebuild makefile

When selected, the makefile will be re-built for every build.

#### **Cobol program extension to compile**

File extensions that the compiler will recognize as source files, and compile in a BUILD.





#### Cobol program to preprocess before compile

File extensions that will be precompiled when using the -preprocess compiler flag.

#### **Setup Script**

The COBOL-IT setup script.

#### Working directory

Default is project directory.

## **Remote Source Control**

Allows settings for CVS, SVN source code control. This funcitonality is only supported when using the Remote System Explorer.

Remote 9	Source Co	ontrol	$\diamond \bullet \bullet \bullet$		
🗹 Enable	source set	ings	Use external make		
• cvs		⊖ svn	() Other		
Commar	nds				
Add	cvs add				
Commit	cvs comr	nit			
Update	cvs upda	te			
Remove	cvs remo	ve			
Lock	cvs edit				
Unlock	cvs uned	t			
History	cvs log				
Diff					
Check Ou	t/Export	cvs co			
Diff Temp	Folder	.cobol.tmp.revision			
Execution	directory	%р			
Module R	eference	%f			

## **Standard Options**

Standard compiler options. In our examples, we have set -o to .\object and -t to .\lis

Standard Options		⇔ ▼ ⇔ ▼ ▼
Enable source settings	Use external make	
Write compiler errors/warnings to <file> ( -err <file> )</file></file>		<u>B</u> rowse
Save object file in source folder or in <directory> (-o &lt;</directory>	srcdir> or <directory>)</directory>	<u>B</u> rowse
Generate list file in source folder or in <directory> (-t &lt;</directory>	srcdir> or <directory>)</directory>	<u>B</u> rowse
Truncate listing output at column 76 (-ftruncate-listing) Generate verbose display (-v)		
Save intermediate files in source folder or in <directory< td=""><td>&gt; ( -save-temps(=<directory>) )</directory></td><td>Browse</td></directory<>	> ( -save-temps(= <directory>) )</directory>	Browse
Generate thread-safe code ( -fthread-safe )		

#### -err <file>

Causes errors and warnings to be written to <file> instead of stderr.





#### -o <file> | <dir>

Causes a compiled object to be output into <file> or <directory>.

#### -t <file> | <dir>

Causes a program listing to be output into <file> or <directory>.

#### -ftruncate-listing

Causes output of the -t <file> compiler flag to be truncated at column 76.

#### -V

Produces verbose output.

#### -save-temps=<directory>

Causes all intermediate files to be preserved.

#### -fthread-safe

Generates thread-safe executables.

#### Warnings

Configure how the compiler generates warnings.



-w Disables all warnings.

-Wall





Version 2.0

Enables all warnings.

-Warchaic Warns if archaic features are used.

-Wbdb Warns about bdb license. (default).

**-Wcall-params** Warns if non 01/77 items are used for CALL parameters (NOT set with Wall).

-Wconstant Warns if inconsistent constant is used.

-Wimplicit-define Warns of implicitly defined data items.

-Winformation Warns information about 'not recommended' code. –Winformation is applied by default.

-Wlinkage Warns of dangling LINKAGE items. (NOT set with -Wall).

-Wobsolete Warns if obsolete features are used.

-Wparentheses

Warns of lack of parentheses around AND within OR.

## -Wredefinition

Warns if incompatible redefinition of data items are used.

-Wstrict-typing

Warns of type mismatch strictly.

-Wsuggestion Warns suggestions about 'not recommended' code. –Wsuggestion is applied by default.

-Wterminator Warns of lack of scope terminator, such as END-XXX. (NOT set with -Wall)

-Wtruncate Warns of possible field truncations. –Wtruncate is applied by default.

-Wunreachable Warns of unreachable statements.





# **Building and Running hello.cbl**

In our sample project, the compiler configuration is designed to:

- Set the -o compiler flag to create object files to be created in the .\object folder.
- Set the -t compiler flag to create listing files the .\lst folder.
- Set the -I compiler flag to locate copy files in the .\copy folder.
- Set the -g compiler flag to cause debugging information to be added to the object files, so they can be run in debug mode.

Our Runtime configuration should:

- Set the COB\_LIBRARY\_PATH to allow the runtime to locate the compiled object.
- Set the COB\_FILE\_PATH to cause the runtime to locate files in the .\data folder.

# **Project>Properties>COBOL Properties**

Click on the Project folder in the Navigator Window to select it. Right click, and select "Properties" from the right-click dropdown menu to open the Properties dialog window.

In the menu on the left, select "COBOL Properties". In the COBOL Properties dialog, click "Enable source settings" to enable the setting of compiler flags through the Compiler Properties sub-headings in the panel on the left.

	♦ • 0 • • • • • • • • • • • • • • • • •					0	Quick Access	8 8 4
Navigator 🕸	□     ■     hello.cbl ≅     B myfile.sl     B myfile.fd       □     ●     ●     1     IDENTIFICATION DIVI	SION.			~	B Outline      IDENTIFICATION	P	la ∝ ⇔ c
project1 ,launc	New	>	Properties for project1			hello     following constants		
🛩 🗁 сору	Go Into Open in New Window		type filter text		COBOL Properties	INPUT-OUTPUT		
in my ii my li x the data	© Copy Poste > Poste > Delete Mann		<ul> <li>COBOL Properties Accept/Display</li> </ul>	^	Enable source settings	V DATA FILE		
in my 1			CALL Parameters Cmd Flags Customize Setup > Data Handling > DevOps Tools > Dialects			O PROCEDURE     main     write-file     read-file		
🖹 hell	Rename	N. 2						
× ≫ objec ■ hell	a Import a Export							
iii hell iii hell iii hell € hello.	Build Project Refresh Cloge Project Close Unrelated Projects		<ul> <li>Environment</li> <li>File Handling Informational</li> <li>Link</li> <li>Optimize Options</li> </ul>					
	Show in Remote Systems view Run As	>	Pre-processing Options Project Options Bemote Source Control					
	Restore from Local History prd	prd I	Standard Ontions		Restore Defaults Apply			
		3			OK Cancel		8 2 6 -	<b>-</b>
	Team	5						
	Compare With	>						1
	Configure	>						
	Properties							
	Remove from Context Ctrl+Alt+Shift+Dow	n l						

- On the Standard Options screen, set the -o compiler flag to .\object.
- On the Standard Options screen, set the -t compiler flag to.\lst.
- On the Environment>Copy Files screen, set the -I compiler flag to .\copy.
- On the DevOps Tools>Degugging Tools screen, set -g compiler flag.

Click Apply, and then click OK to save your selections, and close the Properties dialog window.





# Standardize a Run Configuration

When you select the "Run" function from the main toolbar, or from the main menubar, the standard Developer Studio Behavior is to create and exec script, and then run it.

### A sample exec script for hello.dll

CALL %COBOLITDIR%\setenv\_cobolit.bat cmd.exe /c start /wait cobcrun hello

The exec script first runs a "Pre-run script", which by default is set to setenv\_cobolit.bat, and then runs the program you wish to execute. The Pre-run script can be customized through the Customize Setup dialog screen.

If all of the programs in your project can be run with the same environment variables, then it is best to set the environment variables in your Pre-run script. In an example in which a program, hello.cbl, has been compiled with the –o .\object compiler flag, the runtime requires that the environment variable COB\_LIBRARY\_PATH be set to .\object. A standardized run configuration would include a Pre-run script that contained the instruction:

(Windows) SET COB\_LIBRARY\_PATH=.\object (Linux) export COB\_LIBRARY\_PATH=./object

# **Customize a Run Configuration**

You can accept the default Pre-run Script, which is the COBOL-IT setup script, and then include environment variable settings in a Run Configuration.

To create a customized Run Configuration for hello.cbl:

Right-click on the COBOL program hello.cbl in the Navigator Window.

Select Run As.... from the right-click dropdown menu, and then select "Run Configuration..." to open the Run Configuration dialog window.



	COBOL-I	Совог-і	T Develope The	r Studio– Gettii Basics	ng Started	Version 2.0
Workspace - C File Edit Source Navigator II Provide Composition Provide Composition Pr	COBOL-IT Developer Studio - project1/hello.cbl - COBOL-IT Developer Studio - project Run Window Help ♥ • 0 • Q • () Ø Ø • () ♥ Ø ↓ ♥ Ø Ø • • • • • © el hello.cbl # @ myfileal @ myfileal   ● %   ♥ • 0 1   DENTIFICATION DIVISION 2   PROGRAM-ID. hello. 3   ENTROMEENT DIVISION 4   INPUT-OUTPUT SECTION. 5   FILE-CONTROL.	Hoper Studio - C\COBOL\COBOLI\Dev.	Studio200\workspace		Outlek Access	X
> 🗠 lst > 🗠 object 😰 hello.cb	o     COPY     myrile.si       New     >       Open     >       Open With     >       E copy        ■ Daste     >       W Delete     Moye       Rename		Run Configurations Create, manage, and ru	In configurations	O DADA     O FILE     O WORKING-STORAGE     X	
	Import     Export     Refresh     Show in Remote Systems view     Run As     Debug As     Cobol     Source Central	해 1 COBOL Run Configurations	v i i Cobol Program i i hello.cbl	● Main \	ronment © Common Browse Browse	
project1/hallo.ct	source control         >           Team         >           Compare With         >           Replace With         >           Ø GitHub         >           Properties            Remove from Context         Ctrl+Alt+Shift+Down           Mark as Landmark         Ctrl+Alt+Shift+Down		Filter matched 2 of 2 ite		Reyert Apply Bun Close	~

# The Run Configuration dialog window

The Run Configurations dialog window consists of a tree structure on the left panel, and a multtabbed window in the right panel.

# Create a new Runtime Configuration, and Assign it a Name

Select "COBOL Program" and click on the "New" toolbar button above the tree structure on the left panel. On the Run Configurations dialog window, change the Name from "New Configuration" to "hello.cbl". Click on the "Apply" button to change the name of the configuration in the tree structure on the left.





Run Configurations		×				
Create, manage, and run configurations						
Image: Second secon	Name: hello.cbl	ion				
Inelio.col	project1	Browse				
	hello.cbl	Browse				
Filter matched 2 of 2 iter	Re <u>v</u> ert	Apply				
(?)	Run	Close				

# Set runtime environment variables on the Environment tab

The Environment tab is where we will set the runtime environment variables that we need to produce the desired behaviors. On the environment tab, click on the "New" button to open the "New Environment Variable" dialog screen. Create new environment entries for COB\_LIBRARY\_PATH and COB\_FILE\_PATH.

Click Apply, and Close, to save your settings, and close the Run Configuration Dialog Screen.

Run Configurations				×
Create, manage, and	run configurations			
	Name: hello.cbl			
	 @ Main №= Runtime ₺ So	ource 📧 Envire	onment 🔲 Commo	n
∽ 🖾 Cobol Program	Environment variables to	) <u>s</u> et:		
leilo.cbl	Variable	Value		N <u>e</u> w
	COB_FILE_PATH	.\data		Select
	COB_LIBRARY_PATH	.\object		Edit
				E <u>u</u> it
				Kem <u>o</u> ve
	<u>A</u> ppend environmen <u>Replace</u> native enviro	t to native env onment with sp	vironment pecified environment	
			Revert	Apply
Filter matched 2 of 2 iter				, 444.Y
?			<u>R</u> un	Close





We are now ready to review the Clean, Build, Run, and Debug operations.

# Compile, Run, Debug

# **Key Concepts**

## Main Toolbar

Project>Build All and Project>Build Project can be used to Compile, and not Run a Program

A Build can be "forced" by using the "Clean" function before running any of the "Build", "Run", or "Debug" functions.

The Run toolbar button on the Developer Studio main toolbar builds (if necessary) and runs the program.

#### Compiler console window

Results of the Compile can be checked in the Compiler Console Window.

# Clean, then Build

Select Project from the Main Menu, and then Clean... from the dropdown menu to open the Clean dialog screen.

Workspace - COBOL-IT Developer Studio - project1/hello.cbl - COB	DL-IT Developer Studio - C:\COBOL\COBOLIT\DevStudio200\workspace		- 🛛 ×
File Edit Source Navigate Search Project Run Window Help			
🗂 🔻 ( Open Project 🛛 🗎 🖗 🖛 🖗 🔶 💌	≎ <del>▼</del>		Quick Access 📑 🖬 😹 🎋
Se Nav Close Project In myfile.sl In myfile	fd	🖱 🗖 🐉 Outline 🕮	\$ <mark> 4</mark> z ∨ ⊂ □
Build All Ctrl+B IDENTIFICATION D	IVISION.	∧ ✓ ◎ IDENTIFI	CATION
✓	10.	<ul> <li>hello</li> </ul>	
Build Working Set > ENVIRONMENT DIVITAGE	SION.	V O ENVIRON	NMENT
Clean FILE-CONTROL.	🐼 Clean — 🗆 🗙	INPUT	T-OUTPUT
Build Automatically COPY "myfile		V 🛛 DATA	
Properties	Clean will discard all build problems and built states. The next time a build occurs the projects	FILE	WING CTODAGE
FILE SECTION.	will be rebuilt from scratch.	> WOR	KING-STORAGE
10 COPY "myfile.fd"	Clean all projects     Clean projects selected below	* • PROCED	URE
12 WORKING-STORAGE	⊡⊜project1	• main	file
13 77 file-status		• read-	file
14 77 ws-dummy PIC			
16 PROCEDURE DIVIS			
17 main.			
18 OPEN OUTPUT			
20 CLOSE myfile			
21 *			
22 OPEN INPUT r	Start a build immediately		
23 PERFORM read	Build the entire workspace		
25 *	Build only the selected projects		
26 DISPLAY mvf	OK Cancel	>	
🗷 Problems 🗳 Console 😂 🖷 Progre	S		ik 🖬 🕪 🛃 🐨 🕶 🗂 🕶 🗆
COBOL-IT Compiler			
			~
			~
<			>
project1/hello.cbl			

# The Clean Dialog Screen

Retain the defaults, which will "Clean all projects", "Start a build immediately", and "Build the entire workspace.".





Note- When you have multiple projects in your Workspace, it may be convenient to only build selected projects.

The Clean will force a compile of the Project. The Build that is executed automatically is a compile of all the programs in the project, using the compiler flags set earlier in this exercise.

Clean and Build does not trigger a Run of the program.

# Clean>Build updates the COBOL-IT Compiler console.

The COBOL-IT Compiler console window is updated with the results of the Build. In between informational messages that reference the Build function, you will see the command-line that is being executed, with cobc, the compiler flags that are being used, and the program name.

When the compile is successful, as in the example below, you will see the name of the "C" program that was successfully compiled by the host "C" compiler, and information on intermediate files that were created.

When the compile is not successful, you will see compiler error reports, as they would appear if you had executed the compile command on the command line. We will examine this case more closely later in this document.



# Clean, Build, and Rebuild a single program

Right click on a program in the Navigator window, and select Cobol>Clean to clean a single program. Select Cobol>Build to build a single program. Select Cobol>Rebuild to rebuild a single





#### program. Use these functions when you are working on a single program in a larger project.

Eila Edit Source	COBOL-IT Developer Studio - projec	t1/hello.cbl - COBOL-IT Develo	oper Studio - C:\COBOL\COBOLIT\[	DevStudio200\worksp	pace			- 0	X
	★ ▼ 0 ▼ 4 ▼ 10 A ▼ 10 12	▼ § ▼ \$ \$ \$ \$ \$ \$ \$					C	uick Access	12 📓 🏘
<ul> <li>Navigator ≅</li> <li>Navigator ≅</li> <li>⊕ Qie Qie Qie</li> <li>Barnches</li> <li>≥ copy</li> <li>≥ data</li> <li>≥ lst</li> <li>&gt; ≥ object</li> <li>c hello.cbl</li> </ul>	* • • • • • • • • • • • • • • • • • • •	<pre>view view view view view view view view</pre>				•	COUTINE CONTRIBUTION CONTRIBUT	uick Access   1 a	
	Paste     Ageste     Ageste     Ageste     Moxe     Rename     Import     Export     Refresh     Show in Remote Systems view     Bun As     Dehun As	, , ,					• read-tile		
	Cobol	>	Build			>			
	Source Control Team Comp <u>a</u> re With Paplace With	>	Clean Rebuild Show Build Command				B, AT	¥ 💽 🕶 🕇	• • •
	GitHub     Properties	> >							~
project1/hello.ct	Mark as Landmark	Ctrl+Alt+Shift+Down Ctrl+Alt+Shift+Up							,

Use the Show Build Command to review the build command prior to executing the build:

🞯 Build command		×
Build command based on the compiler preferences:		
kobc hello.cbl -l ".\\copy" -g -fixed -fsign-ascii -t ".\\lst" -m -o ".\\object" -MT " \\object\\hello.dll" -MF .cobolit.tmp.10.0.0.153\hello- ee893c64be5420f0387fd5eb40475ed35c416450.d		
	ОК	

# **Open With...>Text Editor to view text files**

In this example, we have created a list file, called "hello.lst" in the lst folder. The list file is a text file.

To open hello.lst in an Editor window, single-click hello.lst to select it, then right-click, and select Open With>Text Editor.



	COBOL-IT	COBOL-IT Developer Studio– Getting Started The Basics Version 2.0
<ul> <li>workspace - CC</li> <li>File Edit Source</li> <li>▼ ■ ■ ■ ■</li> <li>Navigator ≅</li> <li>⊕ □ ⊕ = 0</li> <li>Porject1</li> <li>● Jaunches</li> <li>&gt; ⊕ copy</li> <li>&gt; ⊕ data</li> <li>&gt; e lst</li> <li>■ hello.lst</li> <li>&gt; object</li> <li>€ hello.cbl</li> </ul>	DBOL-IT Developer Studio - project1/hello.cbl - COBOL-IT Dev Navigate Search Project Run Window Help	Noper Studio - C\COBOL\COB
	Source Control	
	Tgam Compare With Replace With © GitHub	
	Remove from Context     Ctrl+Alt+Shift+Down	
project1/lst/hello.l	d Mark as Landmark Ctrl+Alt+Shift+Up	

# **Running the COBOL program**

To Run hello.cbl, first open the file in the Code Editor. This places focus on the source file in the

Navigator view, and makes the Run button on the main toolbar visible. Then, click on the Run button to run hello.cbl.







# **Running in the Debugger Perspective**

You are now ready to Run the COBOL program in the Debugger Perspective.

[x] We included the -g compiler flag in our compile string, and

[x] Our Window>Preferences>Run/Debug>Perspectives dialog has been set to open the Debugger Perspective when we launch the debugger.

To Run hello.cbl in the Debugger Perspective, first open the file in the Code Editor. This places focus on the source file in the Navigator view, and makes the Debug button on the main toolbar

visible. Then, click on the 🎽 Debug button on the Developer Studio toolbar.

This will cause the Debugger Perspective to open and run hello.cbl.

🔘 workspace	- Debug - project1/hello.cbl - COBOL-IT Develo	er Studio - C:\COBOL\COBOLIT\DevStudio200\workspace	-	- C	×	ζ
<u>File Edit So</u>	urce <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>W</u> indow	Help				
📑 🗕 🖻 🙆	``````````````````````````````````````	えのぶ 気変    ◆ ◆ ◆ ◆ ◆ ◆	Quic	k Access	2 🛛	*
‡ Debug ∺		C:\Cobol\CobolIT\BIN\cobcrun.exe	- 🗆	$\times$	9 🗸 🖷	
✓ I hello.cb	[Cobol Program]					
J /C:/C	DBOL/COBOLIT/DevStudio200/workspace/projec					
✓ M hello.cb	[Cobol Program]					
Y 🕫 Cobo	niop[0]					
	nello () line: 17					
J /C:/C	DBOL/COBOLIT/DevStudio200/workspace/projec					Û
<					>	
🖬 hello.cbl 🕸	🖻 myfile.sl 📄 myfile.fd				a_ ⊽ ==	
14	77 ws-dummy PIC x.	hello world!				^
15					1	
• 17	main.				1	
18	OPEN OUTPUT myfile.					
20	CLOSE myfile.				1	
21	*					
22	OPEN INPUT myfile.				1	
23	CLOSE myfile.					
<	-					~
Console 🛙	a Tasks				3	
COBOL-IT Co	mpiler					
						$\sim$
						$\sim$
		Writable Smart Insert 17 : 1				

Single step through the STOP RUN command to terminate the program.

# **Toggle Perspectives to re-enter the Developer Studio Perspective**

After terminating a debug session, you can toggle back to the Developer Studio Perspective, and resume development. You can widen the tab area that displays available perspectives by dragging it to the left. When you have widened the tab, you will see the two perspectives, COBOL-IT Debugger Perspective and COBOL-IT Developer Studio Perspective. To toggle between the two, just click on from one tab to the other.



CO	BC	DL-IT <sup>co</sup>	BOL-IT	Develope The	r Studio Basics	– Gettin	g Started	Version 2.0
© workspace - COBOL-IT Developer Studio - File Edit Source Navigate Search Project ○ Navigator ☆ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○	Project1/hello. Ren Window 2 ♀ ↓ ↓ ↓ 10 11 11 12 13 14 15 16 17 17 18 19 20 21 22 23 24 25 26 27 29 30 31 32 33 34 35 36 37	<pre>cbl-COBOL-IT Developer Studio - CACOBO / Help</pre>	ILCOBOLIT\DevStudi	o200\workspace		•	OUICH ACCO OUICH ACCO OUICH ATION OUICH ATION OUICH ATION OUICH ATION OUICH ACCO OUICH	55
	E Problem	s ™ © Console ≔Progress				>	p. = = t	-
	0 items							
	Description	1	Resource	Path	Location	Туре		
				14/2.11	c			

# **Correcting compiler errors from the Problems View**

Compiler errors are recorded on the Problems View, which records program name, and line number, and provides a clickable interface to go to the error. In the example below, a variable has been misspelled, in the statement ACCEPT w-dummy. In the Problems View, click on the red (x) next to the informational message "w-dummy undefined" to select it, and then double-click, and the error line is located.

You will notice that the error line in the source has also been marked with a red (x), and that the line in which the error is located has been highlighted in blue.



	30	L-IT	COBOL-IT	Deve	loper Sto The Ba	udio– Gettir sics	ng Start	ed	Ņ	/ersio	on 2.0
<ul> <li>Workspace - COBOL-IT Developer Studio</li> <li>File Edit Source Navigate Search Project</li> <li>Navigator ::</li> <li>Project1</li> <li>Jaunches</li> <li>© copy</li> <li>© data</li> <li>© bellolst</li> <li>© object</li> <li>€ hello.cbl</li> </ul>	- project1/hello. t Run Window 0 9 9 9 9 9 1 10 10 1 10 10 1 10 10 1 11 1 12 1 13 1 14 15 16 1 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 3 32 33 34 35	bd - COBOL-IT Developer S Help COPY "myfile.fd". COPY "myfile.fd". WORKING-STORAGE S 77 file-status PI 77 w-dummy PIC x. PROCEDURE DIVISIO main. OPEN OUTPUT m PERFORM write CLOSE myfile. DISPLAY myfil ACCEPT ws-dum STOP RUN. write-file. MOVE "weold!" WRITE myfile- read-file.	d ectION. C xx. N. yfile. -file. file. file. file. TO first-word. TO first-word. TO second-word. record.	COL 10.	5200\workspace			Control Contr	ITIFICATION ello IRONMENT NPUT-OUTPUT A ILE VORKING-STORA- VORKING-STORA- VORKING-STORA- VORKING-STORA- in- in- in- tile-status v dummy CEDURE aain <i>ri</i> te-file ead-file	5E	X X Access
	37	READ myfile N	EXT RECORD.				>	-			
	Problems	🗴 🗉 Console 🖷 Progress								51	~
	2 errors, 0 v	arnings, 0 others									
	Description	^		Resource	Path	Location	Туре				^
	✓ 9 Errors	(2 items)									
	'ws	-dummy' undefined		hello.cbl	/project1	C:/COBOL/COB.	COBOL				
	O Con	pilation error		project1		Unknown	COBOL				~
	Con				Writable	Smart Insert 27	43				

# Correcting compiler errors from the Compiler Console View

Compiler errors are also recorded on the Compiler Console View, as follows:

C:/COBOL/CobolIT/train/project1/hello.cbl: In paragraph 'main': C:/COBOL/CobolIT/train/project1/hello.cbl:27: Error: 'ws-dummy' undefined

This output is not in a clickable interface. However, you can easily isolate the line number in which the error occurred, and use the Go To Line Number (Ctl + L) function in the Code Editor to locate the error.

🞯 Go to Line	×
Enter line number (137):	
27	
ОК	Cancel

After correcting a compiler error After correcting a compiler error, you should:

>SAVE your changes >CLEAN >BUILD, ( or REBUILD, from the Project Cobol>Rebuild function) >Verify that the problem has been corrected in the Compiler Console Window and>or the Problems Window.





Version 2.0

>Run to validate that the problem is fixed.







# Summarizing key features of the Developer Studio

In Summary, Key features of the Developer Studio include:

#### Desktop icons which cause setup batch files to execute.

The setup batch files ensure that the "C" compiler, and COBOL-IT Compiler Suite executables can all be located by the Developer Studio.

#### New COBOL Project and New COBOL Program Wizards

We stepped through the use of the New Project, New Folder and New COBOL Program wizards

#### **IDE** Configuration Interfaces

We reviewed important IDE Configuration interfaces in the Window>Preferences>General, Window>Preferences>Run/Debug, and Window>Preferences>COBOL dialog screens.

#### **COBOL-Aware Code Editor**

We created a small COBOL program, and observed how the Outline Window could be used to facilitate navigation within a COBOL program. We SAVE'd our program in the Project folder.

#### Setting Compiler flags at the Workspace, or Project Level

We set compiler options that would allow us to store compiled objects in the .\objects folder ( -o), to store list files in the .\lst folder (-t), and to compile for use with the Debugger Perspective at the Workspace Level.

#### Creating Standard or Customized Runtime Configurations

Standard Runtime Configurations can be made by setting environment variables in the COBOL-IT setup script. Runtime Configuration interface to set runtime environment variables We created a runtime configuration for our program hello.cbl, so that the runtime would be able to locate the compiled objects in the .\object directory ( SET COB\_LIBRARY\_PATH).

#### Clean, Build, and check output in the Compiler Console Window

We used the Clean and Build function to compile the program, and then opened the Compiler Console Window to view the results of the Compile.

## Run, and Run in the COBOL-IT Debugger Perspective

We used the Run and Debug functions on the Developer Studio toolbar to run the program, and then run the program in the Debugger Perspective.

#### Compiler errors can be located through a clickable interface in the Problems Window

We deliberately created an error condition in the program, and observed the results in the Problems window, as well as in the Compiler Console Window. We corrected the problem, and ran the program.





# Additional development scenarios

# Using existing source code in its current location

This chapter guides the user through the use of the Developer Studio with existing source.

The chapter presents a practical exercise that includes launching a workspace, creating a new project using existing source code, and developing, compiling, running and debugging the existing COBOL programs.

The "New COBOL Project" wizard with "Use Existing Source" adds project information to the selected directory containing the existing source.

Select New>Project, and then select COBOL>COBOL Project to open the Create COBOL Project Wizard.

In the Create COBOL Project Wizard, enter a new project name, "project2", and select the radio button "Create project from existing source".

# **New Project Using Existing Source**

Select File>New>Project...>COBOL>COBOL Project , and click on [Next] to open the COBOL Project Wizard

workspace - COBOL-IT Developer Studio - pr	roject1/hell	o.cbl - COBOL-IT Developer St	udio - C:\COBOL\COBOLIT\DevStudio200\workspace	– o ×	
File Edit Source Navigate Search Project R	Run Windo	ow Help			
New Alt+	+Shift+N >	COBOL Project		😰 🕸 📓 Quick Access	
Open File		Project	•	🗖 🐉 Outline 🕮 🔅 🦂 🗢 🗖	
Open Projects from File System		COBOL Program		IDENTIFICATION	
Close Close All Ctrl+:	Ctrl+W Shift+W	File		• hello     ENVIRONMENT	
i Save ■. Save As Save All Ctrl+ Revert	Ctrl+S +Shift+S	Ctrl+N PROCEDURE DIVISION main. OPEN OUTPUT my	Create a COBOL project Create a COBOL project in the workspace or in an external location	O INPUT-OUTPUT     O DATA     O FILE     ✓ ◎ WORKING-STORAGE	
Move  Rename  Refresh	F2 F5	PERFORM write CLOSE myfile.	Project name: project2 Contents	Browse For Folder	×
Convert Line Delimiters To	>	PERFORM read-	Create new project in workspace		
Print	Ctrl+P	CLOSE myfile.	Create project at existing location (from existing source)	> CIT31029	^
Switch Workspace Restart	>	DISPLAY myfil ACCEPT ws-dummy STOP RUN.	Host: Localhost	> COBOLIT > COBOLIT64	
≧a Import ≧a Export		* write-file. MOVE "hello "	Directory: C\COBOL\cobolsrc Browse	Cobolsrc	
Properties A	Alt+Enter	MOVE "world!"	Working sets	data	
1 hello.cbl [project1] 2 hello.lst [project1/lst]		wRITE myfile-r	Add project to working sets      Wgrking sets      Select	Eolder: cobolsrc	~
4 myfile.sl [project1/copy]		READ myfile N	The specified external location already exists. If a project is created in this location, the wizard v automatically try to detect existing sources and configure the buildpath appropriately.	vill Make New Folder OK	Cancel
Exit	COBOL-IT	s ▣ Console ≋ ≒ Progress Compiler	<ul> <li>wnonwikwy ny w ueter tensing sources and connyure me buncpath appropriately.</li> </ul>		

In the Create a COBOL project Wizard, enter a project name. In the example below, we create a project called project2.





Select the radio button titled "Create project at existing location (from existing source). Browse, and select the sample directory at C:\COBOL\cobolsrc. The new project project2 will be added to the Workspace. Eclipse artifacts, such as the .project file, will be created in the existing directory. Click [Finish].

## Eclipse artifacts are created in the existing directory

Note that eclipse artifacts, such as the .buildpath and .project files have been added to the existing directory as a result of having created a new Project in the existing directory.

C:\COBOL\cobolsrc>dir Volume in drive C is OS Volume Serial Number is F4D5-6238

Directory of C:\COBOL\cobolsrc

12/18/2017	05:46 PM	<dir></dir>	
12/18/2017	05:46 PM	<dir></dir>	
12/18/2017	03:52 PM	106.	buildpath
12/18/2017	05:54 PM	<dir></dir>	.cobolit.tmp.10.0.0.153
12/18/2017	04:44 PM	<dir></dir>	.launches
12/18/2017	03:52 PM	512.	project
12/18/2017	04:37 PM	<dir></dir>	.settings
12/18/2017	05:49 PM	<dir></dir>	С
07/13/2017	02:08 PM	<dir></dir>	сору
12/18/2017	05:53 PM	<dir></dir>	data
12/18/2017	05:47 PM	1,231	holidaysIX.cbl
07/28/2017	09:08 AM	6,260	labfile2.cbl
12/18/2017	05:47 PM	<dir></dir>	lst
12/18/2017	05:49 PM	<dir></dir>	object
4 File(s) 8,109 bytes			
10 Dir(s) 350,442,024,960 bytes free			

Existing files and folders appear in the Navigator Window

If you add or delete folders or files here, they are added/deleted to/from the system folders. If you add or delete files or folders at the system level, they will be added/deleted in your project as well.





#### COBOL-IT Developer Studio– Getting Started The Basics

# **Project>Properties>Resource**

Selec project2 by clicking on it, then right-click, and select Properties from the right-click dropdown menu to verify that location of the project folder is at the location of the existing source:

Properties for project2				$\times$
type filter text	Resource		← ▼ 0	⇒ ▼ ▼
<ul> <li>Resource</li> <li>Linked Resources</li> <li>Resource Filters</li> <li>Builders</li> <li>COBOL Formatter</li> <li>COBOL Properties</li> </ul>	Path:       /project2         Type:       Project         Location:       C:\COBOL\cobolsrc         Last modified:       December 18, 2017 at 3:54:21 PM         Text file encoding			<b>P</b>
Accept/Display CALL Parameters Cmd Flags Customize Setup	<ul> <li>Inherited from container (Cp1252)</li> <li>Other: Cp1252</li> <li>Store the encoding of derived resources separately</li> </ul>			
<ul> <li>&gt; Data Handling</li> <li>&gt; DevOps Tools</li> <li>&gt; Dialects</li> <li>&gt; Environment</li> </ul>	New text file line delimiter <ul> <li>Inherited from container (Windows)</li> <li>Other: Windows </li> </ul>			
<ul> <li>&gt; File Handling Informational</li> <li>&gt; Link</li> <li>Optimize Options</li> <li>Pre-processing Options</li> <li>Project Options</li> <li>Remote Source Control Standard Options</li> <li>Warnings</li> <li>Project References</li> <li>Refactoring History</li> <li>Run/Debug Settings</li> <li>&gt; Task Repository</li> </ul>				
Task Tags WikiText	Restore De	faults	<u>A</u> ppl	У
(?)	ОК		Cancel	





Version 2.0

# Project>Properties>COBOL

Use the Project>Properties>COBOL dialog screen to check, and set compiler flags for this project. The compiler flags for project1 were set at the project level, and are not inherited by project2. Set the –I <directory>, -t <directory>, -o <directory>, and –g compiler flags.

# **Clean and build project2**

As there are now two projects in the Workspace, both the Clean and Build portions of the dialog screen contains options.

In the Clean portion of the dialog screen, select the radio-button titled "Clean projects selected below", and select the checkbox titled "project2".

In the Build portion of the dialog screen, select the radio-button titled "Build only selected projects".

Click OK.







#### COBOL-IT Developer Studio– Getting Started The Basics

Version 2.0

# The Compiler Console Window

The results of the compile are in the Compiler Console Window:



# Create a Run Configuration for project2>holidaysix.cbl

Right-click on holidaysix.cbl, select "Run As" from the dropdown menu, and then select "Run Configurations" from the subsequent dropdown menu. In the Run Configurations wizard, select holidaysIX.cbl in project2 on the Main tab.

Run Configurations		×
Create, manage, and	run configurations	
Image: Second system     Image: Second system       Image: Vype filter text       Image: Second system       Image: Second system <th>Name: holidaysix.cbl  Main Muntime Source Environment Common Project: Program holidaysIX.cbl</th> <th>Browse Browse</th>	Name: holidaysix.cbl  Main Muntime Source Environment Common Project: Program holidaysIX.cbl	Browse Browse
Filter matched 3 of 3 iter	Revert	Apply
?	Bun	Close





Version 2.0

# Add the environment variables

Set the Environment Variables on the Environment tab.

Run Configurations			×		
Create, manage, and	run configurations				
Image: Second secon	Name: holidaysix.cbl Main @•• Runtime & Source EnvironmentCommon				
<ul> <li>✓ I Cobol Program</li> <li>I hello.cbl</li> <li>I holidaysix.cbl</li> </ul>	Environment variables to <u>s</u> et:				
	Variable	Value	N <u>e</u> w		
	COB_FILE_PATH	.\data	Se <u>l</u> ect		
	COB_LIBRARY_PATH	.\object	E <u>d</u> it		
			Rem <u>o</u> ve		
	<u>Append environment to native environment</u> <u>Replace native environment with specified environment</u>				
Filter matched 3 of 3 iter		Re <u>v</u> ert	Apply		
?		Run	Close		

# Run the COBOL program



# **Running the COBOL program in the Debugger Perspective**

Click on the "Debug" button on the Developer Studio toolbar to run holidaysIX in the Debugger Perspective.

Single step with the Step Into (F5) function. Run the program with the Resume (F8) function. Details on the Debugger Perspective are contained the the Debugger Perspective Chapter.



COBOL-IT Developer Studio- Getting Started The Basics			
Image: Search Project2/holidays/X.cbl - COBOL-IT Developer Studio - C:\COBOL\COBOLIT\DevStudio         Ele Edit Source Navigate Search Project Bun Window Help         Image: Search Project Bun Window Help	o200\workspace		
V Deug =      V Deug =	Workables     ** Breakpoints     ** Expressions     ** Intera       Name     *     * WORKING-STORAGE       • RETURN-CODE     *     *       • TALLY     •     SORT-RETURN       • NUMBER-OF-CALL-PARAMETERS     •     numeral-numeric       • numeral-numeric     •     •       • num-records     •     •       • ws-quotient     •     •       • holiday-record-id     •     •       • holiday-status	Value         Image: Console         Image: Console </th	
bolidaystX.cbl # c loadfile2.cpy c loadfile.cpy holidays.fd      procedure division.		Contine **  Conte	

# Importing existing source code into a Project structure

The COBOL Import Wizard provides a way for you to copy existing source into an existing Project folder structure. When you import source into an existing Project, all modifications are made to the copies of the source that have been made in the existing Project locations.

# **The COBOL Import Wizard**

The COBOL Import Wizard requires an existing Project. In this exercise, we will import a COBOL source file from the COBOL-IT sample programs into our project1.

To use the COBOL Import Wizard, right-click on the Project folder, and select Import... from the right-click dropdown menu.

From the Import dialog screen, expand the COBOL folder, and select the COBOL Import Wizard, and then click on the [Next] button.




### COBOL-IT Developer Studio– Getting Started The Basics

Version 2.0

🗑 🕲 🐘 ♥ ▼ Ο ▼ 🂁 ▼ 😂 🖋 ▼ 🖻	] 👔 🔻 🖗 🗶 🏟 🗮 🖒 💌		🔢 🕸 😹 Quick Access
vigator # CORE & E CORE protect Go Into Open in New Window Copy Daste X Delete Moye Rename	<pre>c holidaysIX.cbl # ∈ loadfile2.cc 13 data division 14 file section.</pre>	r i loadfile.cpy ≧ holidays.fd	Outline 3 Value Access Outline 3 P &  Outline 3 Outline
Import     Export     Build Project     Close Project     Close Project     Close Unrelated Projects     Show in Remote Systems view     Bun As     Debug As	iday-re Jummy iday-te ture div EN OUTP Y "los DSE hol SPLAY h SPLAY > tept ws	ail COBOL Import Wixed       ail CoBOL Import Wixed       ail Coursage Session       it       ail Coursage Session       sio       T h       fil       day       (id)       (id)	-
Restore from Local History Cobol Source Control T <u>e</u> am	> P RUN.	RAM. v	
Comp <u>a</u> re With Configure	>		^

S Import	—		×
Select			
Import COBOL Source in your project			1
Coloritor incontruinent			
Select an import wizard:			
type filter text			_
> 🧽 General			
V 🗁 COBOL			
COBOL Import Wizard			
Coverage Session			
🚝 Profiler Session			
> 📂 Git			
> 📂 Git			
> 🗁 Install			
> 🗁 Remote Systems			
> 📂 Run/Debug			
> 🗁 Tasks			
> 🗁 Team			
(?) < Back Next > Finish		Cance	1





### Select the Import Directory

In the Import resources dialog screen, begin by clicking on the Browse.. button, which will open the "Import from directory" dialog screen. In the "Import from directory" dialog, select a directory, and click OK. This will cause the From directory: entry-field to be filled with your choice.

🔘 Import						×
File system Import resources	s from the local file sy	stem.				
From directory:	C:\Cobol\CobolIT\S	amples\custome	r	~	B <u>r</u> owse	
Custo	omer		Customer0	.cbl		
Filter <u>Types</u> Into fo <u>l</u> der: pro	<u>S</u> elect All	<u>D</u> eselect All			Bro <u>w</u> se	
Overwrite ex Create top-lo Advanced >>	isting resources witho evel folder	out warning				
?		< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel	

To select only the file ( in this case customer0.cbl ), just select the checkbox in the file in the panel on the right. Click on the Finish button, to import the files into the existing project.





### Imported files are automatically displayed in the project folder

The selected source file is copied into the existing Project Folder, and displayed in the Navigator Window.



#### **Compiler Configuration**

Source files imported into a Project inherit the compiler flags that have been set with Project>Properties. The program "customer0.cbl" that we have imported can be compiled with existing compiler flags, so no new action needs to be taken.





#### Create a Run Configuration for the new source file(s)

Run Configurations			×
Create, manage, and	run configurations		
Image: Second secon	Name: customer0.cbl	ource Environment Commo	n
Customer0.cbl	Variable	Value	N <u>e</u> w
leine holidaysix.cbl	COB_FILE_PATH     COB_LUBRABY_DATU	.\data	Se <u>l</u> ect
	COB_LIBRARY_PATH	.\object	E <u>d</u> it
			Rem <u>o</u> ve
	<u>Append environmen</u> <u>Replace native environ</u>	t to native environment	
Filter matched 4 of 4 iter		Re <u>v</u> ert	Apply
?		Run	Close

Customer0.cbl requires the COB\_LIBRARY\_PATH and the COB\_FILE\_PATH environment variables be set.

### **Clean and Build**

Clean and Build project1, which now contains the imported source file.

he projects will be rebuilt from scratch.	i built states. The next time a build occurs
Clean all projects	Clean projects selected below
<ul> <li>✓ ▷ project1</li> <li>□ ▷ project2</li> </ul>	





Version 2.0

### **Check the Compiler Console Window**

#### This Build is successful.



### Run

Click on the "Run" button on the Developer Studio toolbar.







#### COBOL-IT Developer Studio– Getting Started The Basics

Version 2.0

### **Run in Debug**

Click on the "Debug" button on the Developer Studio toolbar. Single step through the source with the Step Into (F5) toolbar button.



## Dragging and dropping files and folders into a Project

An alternative way to import files and folders into a Project is to select them, and drag and drop them onto the Project folder in the Navigator window. In the Windows Explorer, selected a file, or a group of files and perform a "drag and drop" operation.

### File and Folder Operation

The File and Folder Operation dialog screen provides a number of options for using existing files and folders. Select the "Copy files and folders" radio-button, and click "OK".





### Navigator Window updated

When the files have been COPY'ed into the Project folder, the Navigator Window is updated.

### Summary

New Projects can be associated with Existing Source through the New Project wizard.

When using Existing Source, all Eclipse file artifacts are added to the existing source directories.

All editing is done on the files in the Existing Source directories

Working on multiple projects within a Workspace is possible, though you will notice how certain screens, such as the Clean and Build screen provide more options when this is the case.

Configurations can be made at the Workspace level from within the Window>Preferences interfaces. These configuration settings are automatically applied to all Projects within a Workspace.

Configurations can be made at the Project level, from within the Project>Properties interfaces. These configuration settings do not transfer from one project to another, as you add projects to the workspace.

Configurations can be made at the Program level, by right-clicking on a program, and selecting Project>Proprties. These configuration settings apply only to the selected program.

Runtime configurations are useful in cases where a program requires a special configuration. However, using Standard Configurations, in which the COBOL-IT setup file is altered to include the necessary runtime environment variable settings is likely to be a more convenient solution.

Drag-and-drop functionality from the Windows Explorer directly into the Project folder in the Developer Studio is supported.



Version 2.0



The Developer Studio automatically launches the Debugger perspective when a debug operation is launched. Make sure that your programs are compiled with –g prior to launching the Debugger perspective.

The Enterprise Edition of the Developer Studio can be used to compile, run, and debug programs in projects on remote machines.





# FAQ

# Using "Views" in Eclipse

### **Key Concepts**

The Eclipse IDE is divided into a number of tabbed interfaces called Views. These Views can be Minimized/Maximized, Opened/Closed, Detached/Attached, Repositioned with Drag and Drop.

Lets consider an example:

In the graphic below, the tabbed interfaces are titled "Navigator", hello.cbl, Outline, Problems, Console, and Progress. These are referred to, respectively, as the Navigator View, Code Editor View, Outline View, Problems View, Console View, and Progress View.

all and space:       Universe Stards       - <t< th=""><th></th><th></th><th></th></t<>			
Bit Bit Spare Byoke Big and Byoke Big         Image: Spare Byoke Byoke Big         Image: Spare Byoke Byoke Big         Image: Spare Byoke Byoke Byoke Big         Image: Spare Byoke B	Workspace - COBOL-IT Developer Studio - pro	ject1/hello.cbl - COBOL-IT Developer Studio	- 🗆 X
Image: State of the	Eile Edit Source Navigate Search Project	<u>Bun</u> <u>Window</u> <u>Help</u>	
Weightor II     Image: State and State	📑 • 🔛 🐚   📾 ! 🏘 • 🔕 • 🂁 • 😥	<b>∛ •</b> [ ∰ • ∰ • ♥ Φ • ⇔ •	Quick Access 🔡 🔛 🔛
Constant 0 const	🗞 Navigator 🕄 👘 🗖	C hello.cbl 23	" 🗆 🔮 Outline 🛛 👘 🖧 🔻 🗖 🗖
COBOL-IT Compler	<ul> <li></li></ul>	1 IDMITIFICATION DIVISION. 2 PRODUCTION DIVISION. 5 GUAL NUTSION 5 GUAL NUTSION 77 Un-dummy PIC X. 70 Un-dummy PIC X. 10 PROCEEND DIVISION. main. 5 STOP RUM. 5 STOP RUM.	<ul> <li>◆ ● Index</li> <li>● Hele</li> <li>● ENVIRONMENT</li> <li>◆ ● WORKNe-STORAGE</li> <li>◆ ● PROCIDUEE</li> <li>● main</li> </ul>
COBOL-IT Compiler		Problems 🔲 Console 🐹 🖏 Progress	Sk 🖬 🚱 🚽 😁 🕶 🗖
		COBOL-IT Compiler	
		۲ 	,



Version 2.0



### Minimize/Restore a View

Minimize / Maximize buttons are located in the upper right-hand corner of each of the Views on your Developer Studio IDE.

🞯 workspace - COBOL-IT Developer Studio - proj	tct1/hello.cbl - COBOL-IT Developer Studio	- 🗆 ×
<u>File Edit Source Navigate Search Project</u>	<u>Run Window Help</u>	
📑 • 🔛 🐚   🐘 🕸 • O • 💁 • 🗁 /	Ø • [約 • 頁 • ゆ ◆ • ♀ •	Quick Access 😰 🔯 🛱 🌞
🗞 Navigator 🕄 👘 🗖	c hello.cbl 23	" 🗖 🔮 Outline 😫 🕼 🐙 🖤 🗖 🗖
C C C C C C C C C C C C C C C C C C C	1 IDENTIFICATION DIVISION. 2 PROCNA-DL hells. 5 UNIX DIVISION. 5 UNIX DIVISION. 5 UNIX DIVISION. 77 UN-Sammy PCC N. 77 UN-Sammy PCC N. 77 UN-Sammy PCC N. 8 UNIX DIVISION. 8 UNIX DIVISION. 8 UNIX DIVISION. 9 UNIX DIVIS	• © DENTRCANDN     • wello     • Nello     • Nello     • Anta     • © Anta     • © WorkINA-STORAGE     • © PROCEMEE     • main
	😰 Problems 📮 Console 🐹 🖷 Progress	B. 🛃 💷 🚽 🖬 🕶 🖻 🖬
	COBOL-IT Compiler	
	с с	× 2

To Minimize the Navigator View, click on the Minimize button in the upper right-hand corner of the Navigator View. The Navigator View minimizes to the left-hand side of the Developer

Studio, and is shown by the icon [15] ( see graphic below ).

COROLIT Developer Funds and all COROLIT Developer Funds	
workspace - COBOL-II Developer Studio - project1/hello.cbl - COBOL-II Developer Studio	- 0 ×
File Fait Fonce Wandate selice Fullet Rn Million Helb	
	Quick Access
e s hello.cbl 🛛	- 🗆 📴 Outline 🕴 🕼 🖓 💆 🗸 🖓 🗖
<pre>CE</pre>	<ul> <li>♥ IBNIFICATION</li> <li>Nello</li> <li>ENRICOMENT</li> <li>♥ DATA</li> <li>♥ ORIXIS-STORAGE</li> <li>♥ PROCEDUE</li> <li>* main</li> </ul>
🕐 Drohlams 🕞 Console 💥 📖 Drogress	
COBOL-IT Compiler	
	· · · · · · · · · · · · · · · · · · ·





To Restore the Navigator Window to its original position, click on the Restore *icon* icon positioned directly above the Navigator icon.

### Maximize/Restore a View

To Maximize the Navigator View, click on the Maximize button in the upper right-hand corner of the Navigator View. The Navigator View occupies the entire screen, and all other views are minimized.

Note that the Toolbar for the maximized Navigator View now has a "Restore" button on it, where the "Maximize" button was formerly placed. Clicking on this "Restore" button restores the desktop.

Note also that the other views that were open have been mimized, and are shown minimized as icons on the right hand side of the Developer Studio, along with "Restore" buttons. The desktop can be partially or fully restored by clicking on these icons.



Clicking on the "Restore" icon above the Code Editor icon restores the Code Editor. Clicking on the "Restore" icon above the Outline View icon restores the entire desktop to the way it was before maximizing the Navigator View.

Clicking on the "Restore" icon at the bottom of the screen restores the entire desktop.

Clicking on any of the minimized View icons restores that View.





#### COBOL-IT Developer Studio– Getting Started The Basics

Version 2.0

### Close/Open a View

To Close the Navigator View, click on the "Close" icon in the right-hand portion of the tab title for the View. When you "Close" a View, it is removed from the Developer Studio Desktop.

i workspace - COBOL-IT Developer Studio - pro	ject1/hello.cbl - COBOL-IT Developer Studio	– 🗆 X
Eile Edit Source Navigate Search Project	<u>B</u> un <u>Window</u> <u>Help</u>	
📑 • 🔛 🐚 🐘 🎋 • 🛈 • 💁 •	<i>┩</i> ▼[約 × ╗ × ७ ゆ ▼ ↔ ▼	Quick Access 😰 🔯 🖽 🎋
%5. Navigator         □           ↓□         ↓□           ↓□ <td>helio cbi      Dentification Division.     PROGRAM-ID. hello.     ENVIRONMENT DIVISION.     ENVIRONMENT DIVISION.     dotta a Division.</td> <td>Outline 🛛 🗊 🛱 🗢 🗆 🔹 IDENTIFICATION • hello • ENVIRONMENT</td>	helio cbi      Dentification Division.     PROGRAM-ID. hello.     ENVIRONMENT DIVISION.     ENVIRONMENT DIVISION.     dotta a Division.	Outline 🛛 🗊 🛱 🗢 🗆 🔹 IDENTIFICATION • hello • ENVIRONMENT
<ul> <li>de data</li> <li>de fat</li> <li>object</li> <li>object</li> <li>observat</li> <li>custome0.dbi</li> <li>fatical</li> <li>project2</li> </ul>	5 WORKING-STORMER SECTION. 6 77 WorkErower Division. 9 Marce Strate St	© DATA > © WORKING-STORAGE > PROCEDURE • main
	Problems 📃 Console 🕴 🖏 Progress	🔍 📓 🔛 🛃 🖬 🕶 😁 📼
	COBOL-IT Compiler	
		Â
	< c	>
project1/hello.cbl		

#### To Open a View

In the graphic below, the Navigator View has been Closed. To Open the Navigator Window, select Window>Show View>Other . Then, in the Show View dialog screen, select General>Navigator, and click "Ok".

🞯 workspa	ce - COBOL-IT Developer Studio - project1/h	ello.cbl - COBOL-IT Develope	r Studio					-			×
Eile Edit	Source Navigate Search Project Run	Window Help New Window	¥				Quick A	ccess	<b>B</b>	0	<b>*</b>
hello.cbl	22	Editor > Appearance >					📴 Outline 🔀	5	$\downarrow^a_{\mathbf{Z}}$	~ -	
1 2	IDENTIFICATION DIVISION. PROGRAM-ID. hello.	Show View >	Other	Alt+Shift+Q, Q		^	<ul> <li>DENTIFICATION</li> <li>hello</li> </ul>				
3 4	ENVIRONMENT DIVISION. DATA DIVISION.	Perspective >					D ENVIRONMENT				
5	WORKING-STORAGE SECTION. 77 ws-dummy PIC x.	Navigation >					> O WORKING-STORAGE				
8	main.	Preferences					<ul> <li>PROCEDURE</li> <li>main</li> </ul>				
٤					,	×					
Problems	Console 🙁 🖷 Progress						🗎 🔐 🚱	1	- 🗅	• -	
COBOL-II Co	mpler										^
<											>





Version 2.0

Show View		×
type filter text		
<ul> <li>&gt; Dynamic Languages</li> <li>&gt; General</li> <li>Bookmarks</li> <li>Classic Search</li> <li>Console</li> <li>Error Log</li> <li>Internal Web Browser</li> <li>Markers</li> </ul>		~
Marigers     Navigetor     Sa. Navigetor     Sa. Vultime     Problems     Progress     Project Explorer     Properties     ✓ Search		*
ОК	Cance	21

# **Managing Multiple Source Folders**

### **Key Concepts**

To set the compiler flags for a Source Folder, right click on the Source Folder, and select Properties>COBOL Properties>Enable Source Settings. Note that when following this set of functions, the Window title for the COBOL Properties dialog window reflects the name of the Source Folder. In the example below, this window title is: "Properties for src1".

1 - B G	🗟 🕶 i 🖨 🛷 🕶 i 🖬 🖗 💀 🗇	• \$ •						Quick Acce	ss 🗄 🗃 👔
File Edit Na ♥ a Gii ♥ Navigator i ♥ Porjecti 2 © copy 2 © data 2 © cobject ♥ © src1 € h ♥ © src2 € α	waysue Search Fright full window hep a • a 4 • a 4 • a 1 • a 4 \bullet		Properties for src1  type filter text  Resource  COBOL Properties  Accept/Display  CALL Parameter  Cmd Flags Customize Setu  Data Handling  Resource  Customize Setu  Data Standing  Customize Setu  Data Standi		e settings yy/include file extension > to copy/include searc	Quick Access B Outline is not available. ×			
	<ul> <li>Refresh</li> <li>Show in Remote Systems view</li> <li>Run As</li> <li>Debug As</li> <li>Restore from Local History</li> <li>Cobol</li> <li>Source Control</li> <li>Tgam</li> <li>Compare With</li> <li>Properties</li> <li>Remove from Context</li> </ul>	Ctrl+Alt+Shift+Down	<ul> <li>&gt; Dialects</li> <li>&gt; Environment</li> <li>Copy Files</li> <li>Encoding</li> <li>Source Files</li> <li>&gt; File Handling</li> <li>Informational</li> <li>&gt; Link</li> <li>Optimize Optio</li> <li>Pre-processing</li> <li>Project Options</li> <li>Remote Source</li> <li>Standard Optio</li> <li>Warnings</li> </ul>	Dialects     Copy Files     Encoding     Source Files     File Handling     Informational     Link     Optimize Option     Pre-processing     Project Options     Remote Source     Standard Optio	\.copy         Add mark for         Search for CO         © Search for CO         Do not sear         Translate COPY         I lower case         upper case         Keep copy s	<ul> <li>Yropertues for stcl</li> <li>type filter text</li> <li>Resource</li> <li>COBOL Properties</li> <li>Accept/Display</li> <li>CALL Parameter</li> <li>Cmd Flags</li> <li>Customize Setu</li> <li>Data Handling</li> <li>DevOps Tools</li> <li>Dialects</li> <li>Environment</li> <li>Copy Files</li> <li>Environment</li> </ul>	Standard Options         Image: Standard Stan	err <file> ) tory&gt; (-o <srcdir> or ectory&gt; (-o <srcdir> or ectory&gt; (-t <srcdir> or cate-listing) n <directory> (-save-t</directory></srcdir></srcdir></srcdir></file>	e external make <directory>) [ <directory>) emps(=<direct< td=""><td>\object .\lst</td></direct<></directory></directory>





- Compiler flag and Environment settings, such as the –I setting in the example above, may use relative notations. Where you see relative notations, the "." refers to the root directory of the Project, not the Source directory.
- In the Workspace we have created for these Getting Started exercises, we have set a compiler flag of –o .\object at the Workspace level. This will cause the object files to be created in the object directory located under the Project directory. This behavior is the same as we saw when the source files were contained in the root Project directory.
- If there were no –o compiler flag setting, the Developer Studio would create the object files in the Source Directory.







# www.cobol-it.com

June, 2020



