

CobolCloud Workbench

Product Brief



CobolCloud is the latest generation of COBOL compiler, built on an open source stack.

CobolCloud is created to preserve COBOL, while enabling applications to adapt to the full range of present environment and future emerging technologies.

CobolCloud Compiler Suite can be configured to work with a wide range of technologies,

and provides access to thousands of options developed to reproduce the behavior of an existing COBOL application.

To support this technology, CobolCloud provides developers with a powerful Workbench that is adaptable to all of the possibilities offered by the multidirectional API.

Integrating the CobolCloud multi-directional API with Microsoft VS Code

The CobolCloud Workbench is integrated with Microsoft VS Code, which is rapidly emerging as the cross-platform, open source development environment of choice. For more information on VS Code, please visit:

<https://code.visualstudio.com/docs>.

The CobolCloud Workbench increases programmer productivity by providing an intuitive way to use the thousands of options of the CobolCloud Compiler. It offers guidance and comfort to the developers by:

- Facilitating the use of CobolCloud's multi-directional API
- Supporting CobolCloud's highly configurable compiler and runtime systems, which have been engineered to support a wide range of dialects and proprietary syntax
- Providing a powerful COBOL debugger which can be integrated into complex enterprise solution stacks
- Enabling CobolCloud's powerful development utilities, such as the Profiling and Code Coverage
- Available in Linux and Windows operating environment, the CobolCloud Workbench runs mission-critical COBOL applications in the most widely used operating environments and in the Cloud.



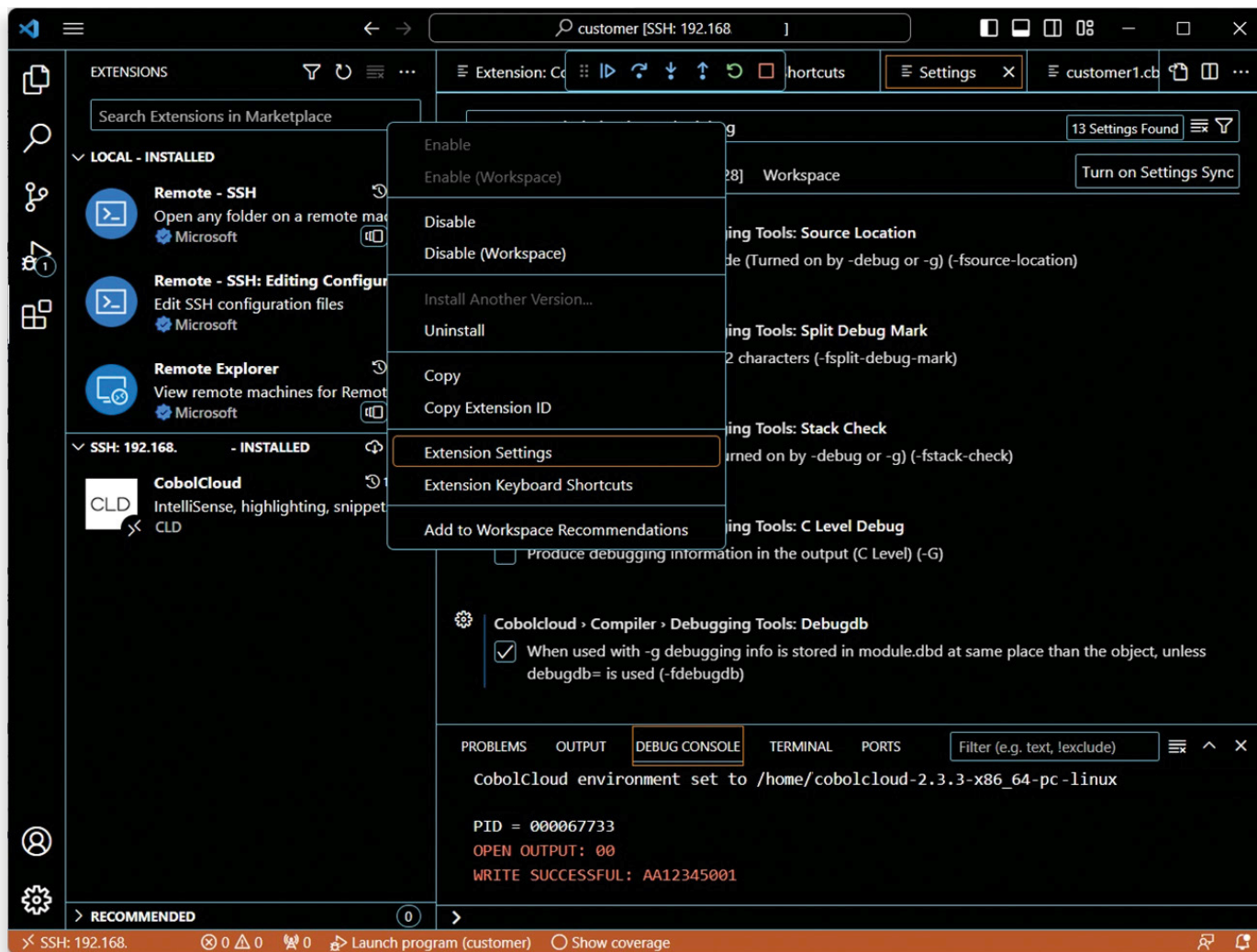
The VS Code work environment is robust, and useful, but it is the proliferation of open source extensions which set it apart from other development environments.
The latest emerging technologies are all represented, including CobolCloud.

The CobolCloud Extension

The CobolCloud Extension provides the means of configuring and executing Compile, Run, and Debug operations, and provides valuable CobolCloud-aware extensions to the COBOL Code Editor.

CobolCloud Configuration Extensions

Workbench Settings



Extension Settings

Allows configuration of the compile, run and debug operations. Includes designating the setup file, setting compiler flags, designating the compiler configuration file. Allows setup for using a pre-compiler and attaching to a database. Enables use of profiling and code coverage utilities and selecting debugging functions such as tracing functions. Also provides for the configuration of the Code Editor which includes colorization, easy navigation within the Code Editor, keyboard shortcuts, and more.

Configure Compiler for use with pre-compiler

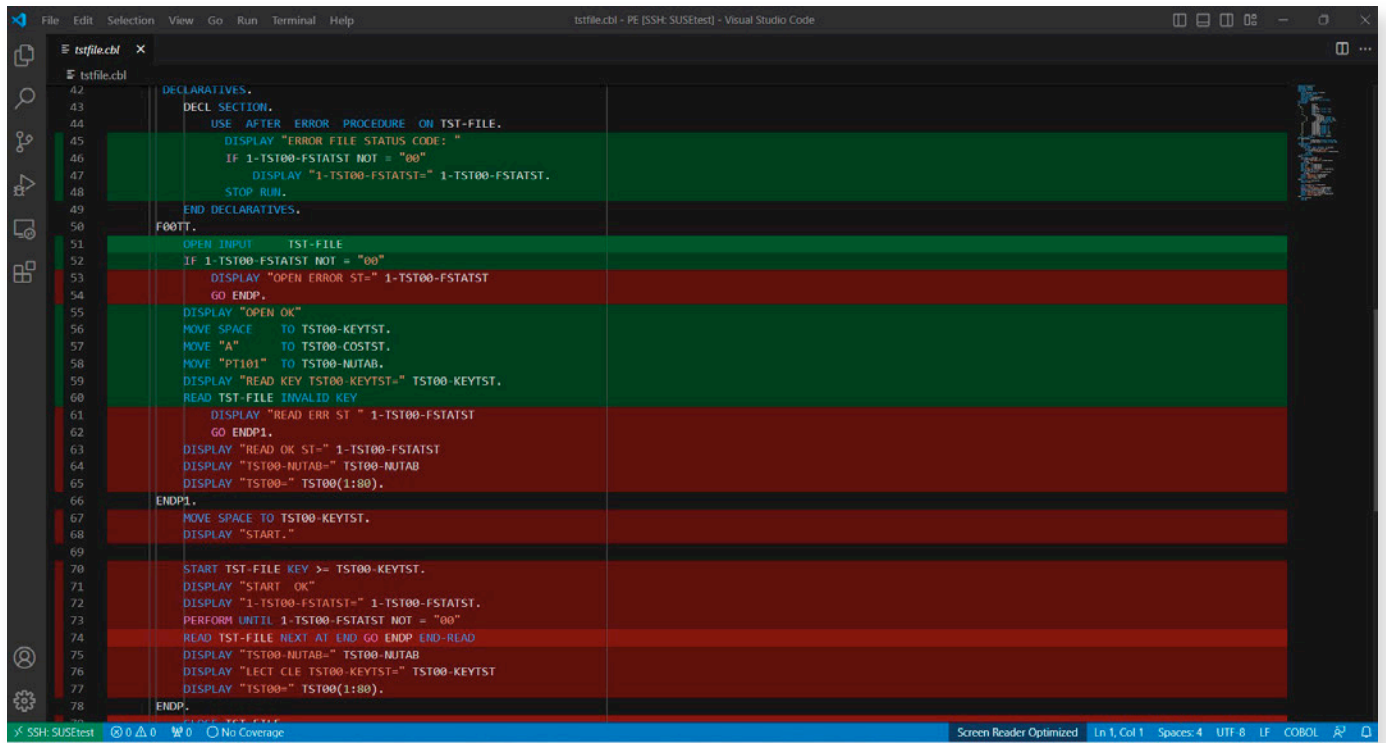
The precompile compiler flag is associated with ascript that takes a source file as input to a precompiler and produces output after precompilation.

Configure Runtime for use with database

Compiled objects can be pre-linked with database libraries at compile-time. Database calls can be configured as static calls using the configuration file.

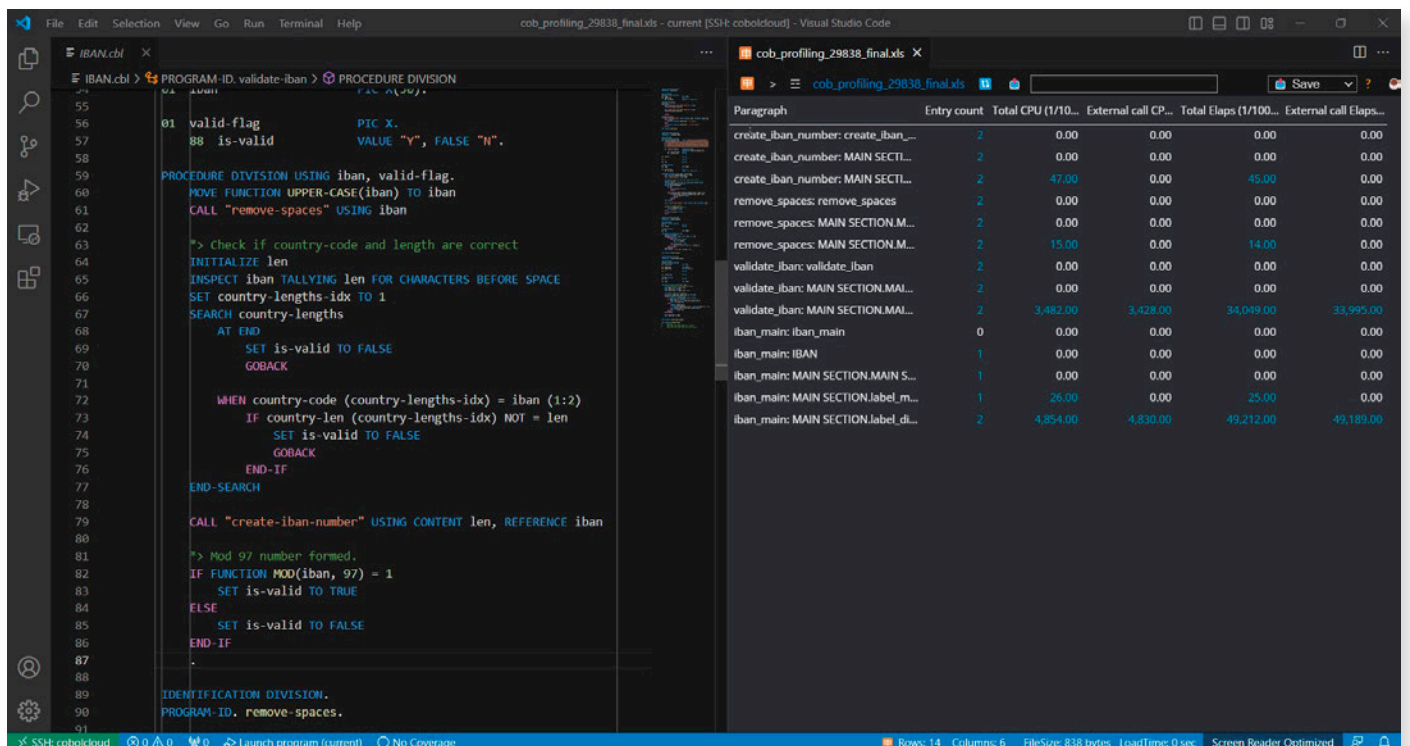
Enable use of Code Coverage Utility

Code coverage settings allow the configuration of the code coverage utility, which reports on code used and not used during a runtime session.



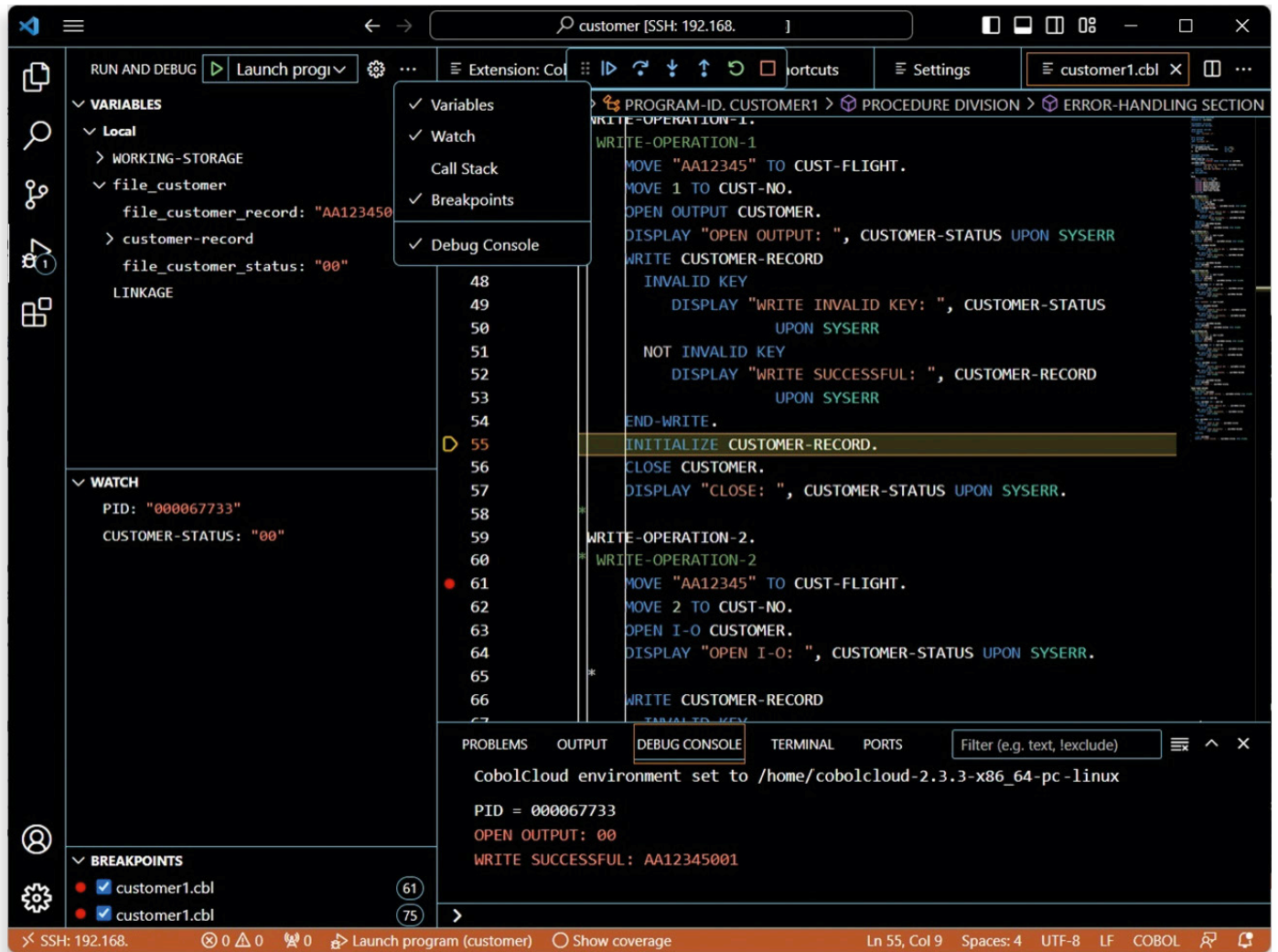
Enable use of Profiling Utility

Profiling settings allow the configuration of the profiling utility which reports on elapsed time and CPU time the runtime spent internally by paragraph/section or externally.



Debugger Extensions

Debugger Views



Variables view

Displays all variables, and current values in the Working-Storage Section, File Section, Linkage Section and Screen Section of the Data Division.

Watch view

In a program with a very large number of Data Division variables, you may wish to turn off the Variables view, and selectively add the variables for which you wish to see the current values displayed. Variables in the Watch view may be used to create "Expressions", such as "CUSTOMER-STATUS=00". An expression will cause the debugger to pause when the condition is met, and the Watch view is updated.

Call Stack view

The Call Stack view is useful in a runtime session in which subprograms are CALL'ed. When operating the debugger, the Call Stack identifies which program CALL'ed the currently running program, and where the CALL was made.

Breakpoints view

The breakpoints view lists the program, and line number within the program that a breakpoint has been set.

Debug Console

The Debug Console shows the displays made to the console when running the debugger.

The Debugger Toolbar

The Debugger Toolbar contains pushbuttons for the debugger functions:

- Continue (F5)
- Step Over (F10)
- Step Into (F11)
- Step Out (Shift+F11)
- Restart (Ctr+Shift+F5)
- Stop (Shift+F5)

Hex to Ascii Hover


Provides Ascii display of variables declared with Hex notation.

Debug Attach

- Debug Attach functionality provides a programmatic way for COBOL programs called from Transaction Monitors, or other programs written in "C" to be paused, and opened in the CobolCloud debugger.

• Code Editor Extensions

Code Editor Keyboard Shortcuts



The screenshot shows the Visual Studio Code interface with the CobolCloud extension settings open. The 'Extension Keyboard Shortcuts' panel is visible, listing various shortcuts for COBOL editing. The 'Keybinding' column shows the key combinations, and the 'When' column shows the contexts where they apply. The 'Source' column indicates that these shortcuts are provided by the CobolCloud extension.

Keybinding	When	Source
Tab	config.coboleditor.enab...	CobolCloud
Ctrl + /	config.coboleditor.comm...	CobolCloud
Ctrl + Alt + D	editorFocus && editorTe...	CobolCloud
Ctrl + Alt + .	editorFocus && editorTe...	CobolCloud
Ctrl + K Ctrl + J	editorFocus && editorTe...	CobolCloud
Ctrl + Alt + P	editorFocus && editorTe...	CobolCloud
Ctrl + Alt + W	editorFocus && editorTe...	CobolCloud
Ctrl + Alt + A	!inSnippetMode && edito...	CobolCloud
Tab	config.coboleditor.enab...	CobolCloud
Ctrl + L	&& config.coboleditor.x...	CobolCloud
Ctrl + Alt + L	!inSnippetMode && edito...	CobolCloud
Ctrl + Alt + ,	editorFocus && editorTe...	CobolCloud
Alt + RightArrow	!inSnippetMode && edito...	CobolCloud
Shift + Tab	config.coboleditor.enab...	CobolCloud
Shift + Alt + A	!inSnippetMode && edito...	CobolCloud
Shift + Alt + C	!inSnippetMode && edito...	CobolCloud
Ctrl + T	config.coboleditor.xedi...	CobolCloud
Ctrl + K	config.coboleditor.xedi...	CobolCloud
Ctrl + M	config.coboleditor.xedi...	CobolCloud
Ctrl + J	config.coboleditor.xedi...	CobolCloud
Ctrl + C	config.coboleditor.xedi...	CobolCloud
Alt + A	config.coboleditor.xedi...	CobolCloud
Ctrl + B	config.coboleditor.xedi...	CobolCloud
Ctrl + E	config.coboleditor.xedi...	CobolCloud
Ctrl + A	config.coboleditor.xedi...	CobolCloud

IntelliSense for keywords

Provides code completion for keywords and intrinsic functions.

Snippets for keywords, intrinsic functions and callable library routines

Provides short code templates for keywords, allowing for the creation of full structures around complex statements such as File IO verbs, Evaluate, If and Evaluate statements. Additionally, provides short code templates for intrinsic functions and callable system libraries.

Keybindings

- Assigns keybindings to dozens of Code Editor behaviors. Keybindings can be customized.
- XEdit'ish keybindings are supported.

Different colors for different code elements

- Assign different colors to keywords, literals, variable names, comments, integers.

Peek at copybooks

Hover over copyfiles to see contents, definitions.

Remove comments

- Allows comments in source code to be removed.
- Rename paragraphs, sections and variables.
- Enables refactoring after changing the name of a paragraph, a section or a variable.

Align storage items

Apply coding standards to alignment of storage items for more readable source code.

Features	Benefits
<ul style="list-style-type: none"> CobolCloud Workbench is available on Linux, Linux and Windows 	<ul style="list-style-type: none"> CobolCloud Workbench is available in the most widely used operating environments, and in the Cloud
<ul style="list-style-type: none"> Integrated with the Microsoft VS Code Workbench 	<ul style="list-style-type: none"> Guarantees access to CobolCloud's latest developments, while providing development teams with access to the most widely used Open Source technologies online and in the Cloud
<ul style="list-style-type: none"> The Workbench may be installed in a Windows platform, and connect with COBOL source files operating in a Linux environment 	<ul style="list-style-type: none"> Integrates with widely used VS Code extensions, including Remote - SSH, allowing users to maintain the advantages of Windows and Linux
<ul style="list-style-type: none"> A powerful set of utilities 	<ul style="list-style-type: none"> CobolCloud Workbench utilities include a CobolCloud-aware Code Editor and Debugger. The Compile and Run operations can be configured to enable Code Coverage and Profiling utilities, to use pre-compilers, and attach to databases
<ul style="list-style-type: none"> CobolCloud extension to VS Code 	<ul style="list-style-type: none"> Allows configuring and executing Compile, Run, and Debug operations, and provides valuable CobolCloud-aware extensions to the COBOL Code Editor
<ul style="list-style-type: none"> Set compiler flags, and designate compiler configuration file 	<ul style="list-style-type: none"> The configuration of the compile, run and debug operations includes setting compiler flags, designating compiler configuration file, enabling profiling and code coverage utilities, and configuring for attachment to a database or for use with a precompiler
<ul style="list-style-type: none"> Configure Compiler for use with pre-compiler 	<ul style="list-style-type: none"> The pre-compile compiler flag is associated with a script that takes a source file as input to a precompiler, and produces output after precompilation
<ul style="list-style-type: none"> Configure Runtime for use with database 	<ul style="list-style-type: none"> Compiled objects can be pre-linked with database libraries at compile-time. Database calls can be configured as static calls using the configuration file
<ul style="list-style-type: none"> Enable use of Code Coverage Utility 	<ul style="list-style-type: none"> Code coverage settings allow the configuration of the code coverage utility, which reports on code used and not used during a runtime session
<ul style="list-style-type: none"> Enable use of Profiling Utility 	<ul style="list-style-type: none"> Profiling settings allow the configuration of the profiling utility which reports on elapsed time and CPU time the runtime spent internally by paragraph/section or externally
<ul style="list-style-type: none"> Debugger Variables View 	<ul style="list-style-type: none"> Variables view Displays all variables, and current values in the Working-Storage Section, File Section, Linkage Section and Screen Section of the Data Division

Features	Benefits
<ul style="list-style-type: none"> • Debugger Watch View 	<ul style="list-style-type: none"> • Creates a small selection of variables for which the current values are displayed. Variables in the Watch view may be used to create "Expressions", such as "variable-name=value". When an expression tests true, the debugger breaks, and the variable is updated
<ul style="list-style-type: none"> • Debugger Call Stack View 	<ul style="list-style-type: none"> • Identifies the stack of CALL's from CALLing programs that have been made in the runtime session prior to the currently running program, and in each case the line number on which the CALL was made
<ul style="list-style-type: none"> • Breakpoints View 	<ul style="list-style-type: none"> • Lists the program, and line number within the program that a breakpoint has been set
<ul style="list-style-type: none"> • Debug Console 	<ul style="list-style-type: none"> • The Debug Console shows the displays made to the console when running the debugger
<ul style="list-style-type: none"> • Debug Attach 	<ul style="list-style-type: none"> • Provides a programmatic way for COBOL programs called from Transaction Monitors, or other programs written in "C" to be paused, and opened in the CobolCloud debugger
<ul style="list-style-type: none"> • Hex to Ascii Hover 	<ul style="list-style-type: none"> • Provides Ascii display of variables declared with Hex notation
<ul style="list-style-type: none"> • Code Editor Keyboard Shortcuts 	<ul style="list-style-type: none"> • Assigns keybindings to dozens of Code Editor behaviors. Keybindings can be customized. • XEdit'ish keybindings are supported
<ul style="list-style-type: none"> • IntelliSense for keywords 	<ul style="list-style-type: none"> • Provides code completion for keywords and intrinsic functions and system library routines
<ul style="list-style-type: none"> • Different colors for different code elements 	<ul style="list-style-type: none"> • Assign different colors to keywords, literals, variable names, comments, integers
<ul style="list-style-type: none"> • Peek at copybooks 	<ul style="list-style-type: none"> • Hover over copyfiles to see contents, definitions
<ul style="list-style-type: none"> • Remove comments 	<ul style="list-style-type: none"> • Allows comments in source code to be removed
<ul style="list-style-type: none"> • Rename paragraphs, sections and variables 	<ul style="list-style-type: none"> • Enables refactoring after changing the name of a paragraph, a section or a variable
<ul style="list-style-type: none"> • Align storage items 	<ul style="list-style-type: none"> • Apply coding standards to alignment of storage items for more readable source code
<ul style="list-style-type: none"> • And many more! 	