isCOBOL^{TM} Evolve

isCOBOL Evolve 2025 Release 2 Overview

isCOBOL Evolve 2025 Release 2 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve: isCOBOL Evolve 2025 R2.

isCOBOL Evolve provides a complete environment for the development, deployment, maintenance, and modernization of COBOL applications.

isCOBOL Evolve 2025 R2 includes multiple improvements in graphical user interface controls.

isCOBOL WebClient 2025 R2 now supports Java 21 and offers new features.

isCOBOL Debugger allows you to dynamically change its look and feel and supports the Dark theme.

isCOBOL IDE and Microsoft Visual Studio extension have been improved too.

Details on these enhancements and updates are included below.

GUI enhancements

IsCOBOL Evolve 2025 R2 integrates a new LAF (visual appearance and user interaction style of applications) named FlatLaf that supports different themes. Several improvements have been implemented in graphical controls: list-box and tree-view controls now support rollover properties, tab-control adds support for multiple hints and close buttons icons, now you can customize the load on demand feature of the grid control, and a new style has been added in the date-entry control.

FlatLaf

The new command line option --flatlaf <theme> for the isrun and isclient commands has been implemented to easily integrate the execution of COBOL applications with the FlatLaf Look & Feel. FlatLaf is a modern open-source cross-platform Look and Feel for Java applications. It looks almost flat, clean, simple, and elegant. FlatLaf comes with Light, Dark, IntelliJ and Darcula themes, that can be activated from the command line, for example:

```
isrun --flatlaf FlatLightLaf PROGNAME
iscrun --flatlaf FlatDarkLaf PROGNAME
isclient --flatlaf FlatIntellijLaf PROGNAME
iscclient --flatlaf FlatDarculaLaf PROGNAME
```

To get the full benefits of this LAF, the application should be executed with the configuration

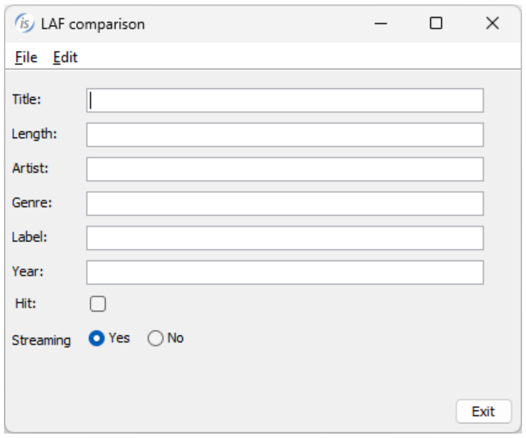
```
iscobol.gui.native_style=1
```

For best results, especially for dark themes, the program should minimally manage the colors of the window, using the J$GETFROMLAF routine to retrieve the theme color, as demonstrated with this code snippet:

```
call "J$GETFROMLAF" using jget-laf-color,
                          "Panel.background",
                          window-background-color
call "J$GETFROMLAF" using jget-laf-color,
                          "Panel.foreground",
                          window-foreground-color
display standard window
        title "LAF comparison"
        background-color window-background-color
        foreground-color window-foreground-color
...
```

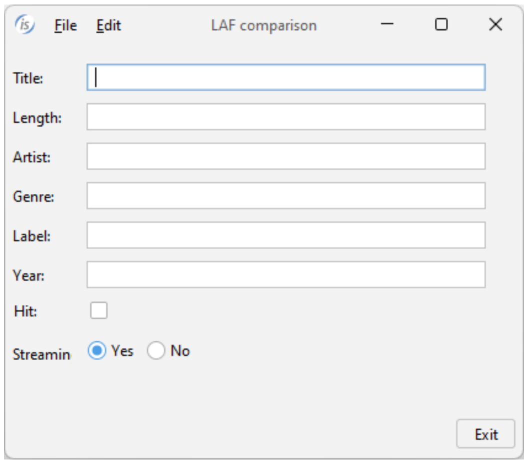
In figures 1 through 3 you can see the same program running with different LAF options to show varying results. In Figure 1, *Windows look,* the program is executed without any option, which defaults to --system being used. Figure 2, *FlatLightLaf shows* the program running using the --flatlaf FlatLightLaf option and in Figure 3, *FlatDarkLaf* the program is executed with the --flatlaf FlatDarkLaf option.

Figure 1. Windows look.



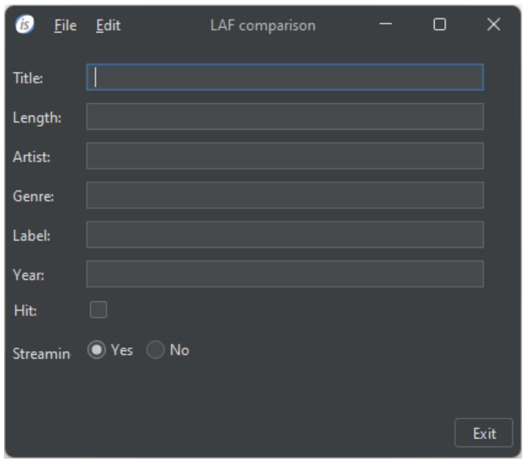
Streaming [✓ Yes] No

Figure 2. FlatLightLaf look.



Streamin [✓ Yes] No

Figure 3. FlatDarkLaf look.



Streamin [✓ Yes] No

Rollover effect

Rollover properties are already available in the grid control to set the rollover row color -- the color that is applied to an item when the mouse hovers over it. In this release the tree-view and list-box controls have been improved and now support the rollover item color. Setting the item-rollover-color property -- or alternatively. the item-rollover-foreground-color and item-rollover-background-color properties -- in the tree-view and list-box controls will result in the item being painted with the specified colors when the mouse hovers over them.

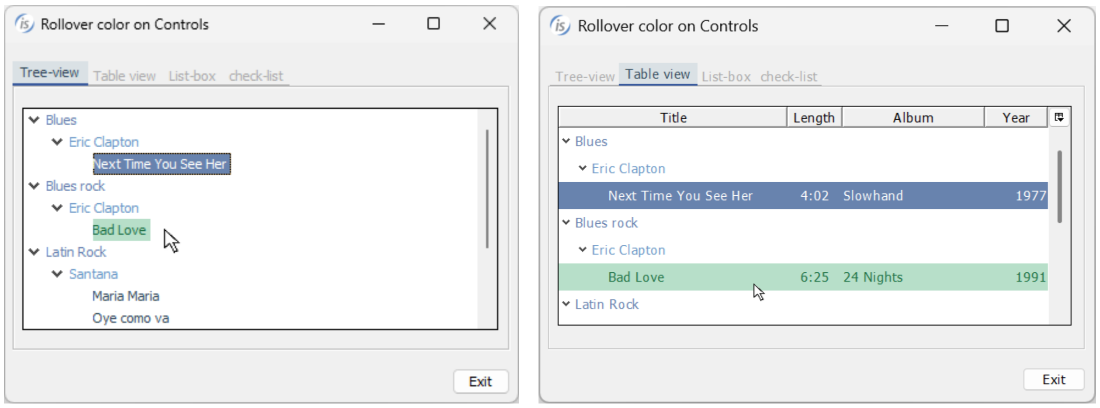
When the mouse hovers in a header cell in the tree-view table-view, the single cell is painted with the color properties specified in the new heading-rollover-color or heading-rollover-background-color and heading-rollover-foreground-color.

The following code shows how to apply the new color properties in the tree-view and list-box controls:

```
05 Tv1 tree-view
   item-rollover-background-color rgb x#B7DFC9
   item-rollover-foreground-color rgb x#217346
   ...
05 Tree-table tree-view table-view
   item-rollover-background-color    rgb x#B7DFC9
   item-rollover-foreground-color    rgb x#217346
   heading-rollover-background-color rgb x#9FD5B7
   heading-rollover-foreground-color rgb x#000000
   ...
05 Ls list-box
   item-rollover-background-color rgb x#B7DFC9
   item-rollover-foreground-color rgb x#217346
   ...
05 Ls-check list-box check-list
   selection-mode lssm-multiple-interval-selection
   item-rollover-background-color rgb x#B7DFC9
   item-rollover-foreground-color rgb x#217346
   ...
```

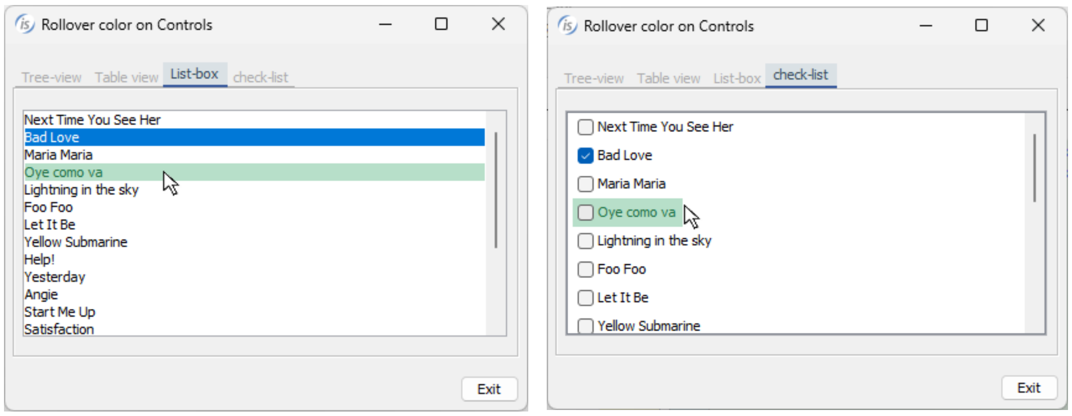
The running program is shown in Figure 4, *Tree-view colors* and Figure 5, *List-box colors*. On both controls, the item highlighted by the mouse pointer is now more noticeable.

Figure 4. Tree-view colors.



Title	Length	Album	Year
Blues			
Eric Clapton			
Next Time You See Her	4:02	Slowhand	1977
Blues rock			
Eric Clapton			
Bad Love	6:25	24 Nights	1991
Latin Rock			

Figure 5. List-box colors.



Tab-Control

The tab-control has been improved by adding the option to set a specific hint for every page. When setting the new property tab-hint in conjunction with the tab-index, the specified hint is shown when the mouse hovers over the tab title area of every style of tab-control: standard, allow-container and accordion.

In addition, a new style called close-buttons adds an "x" icon on the right of every page in the tab, allowing you to manage the closing action of the page. When the user clicks on this icon, the tab page is deleted. The MSG-CLOSE event is fired, and the application can perform actions and block the action if needed.

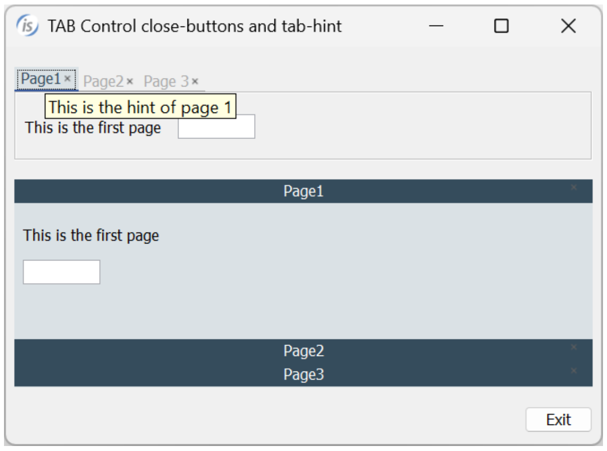
For example, the code snippet:

```
   03 Tb1 tab-control
      line 2 col 2 lines 5 cells size 68 cells
      close-buttons
      ...
   03 Tb2 tab-control accordion
      line 8 col 2 lines 11 cells size 68 cells
      close-buttons
      event tb2-evt
      ...
     modify Tb1 tab-to-add ("Page1", "Page2", "Page 3")
     modify Tb1 tab-index 1 tab-hint "This is the hint of page 1"
     modify Tb1 tab-index 2 tab-hint "This is the hint of page 2"
     modify Tb1 tab-index 3 tab-hint "This is the hint of page 3"
     ...
tb2-evt.
    if event-type = msg-close
       if event-data-1 = 2 and flag-edit = 1
          set event-action to event-action-fail
       else
          move 1 to flag-delete
    end-if
```

creates two tab-controls that have the new close-buttons style, and specific page hints are applied to the first tab-control. The tb2-evt procedure is linked to the second tab-control and it performs a check to cancel the closing of a tab page if a specific condition is not met.

The result of the program running is shown in Figure 6, *Tab-control enhancements*.

Figure 6. Tab-control enhancements.



Grid

The grid-control has a property, LOD-THRESHOLD, that can be used to dynamically load the contents as the user scrolls in the grid, allowing a large number of records to be efficiently shown in the grid. Starting from this release, the MSG-LOAD-ON-DEMAND event has been improved by setting the EVENT-DATA-1 data item to represent the user action that triggered the loading event. The application can use this information to process the event accordingly. Additionally, setting EVENT-ACTION data item to EVENT-ACTION-COMPLETE will prevent the cursor from moving automatically if the action was already performed by code.

For example, the following code snippet:

```
  03 gd grid use-tab
     lod-threshold 80
     event GD-EVT
     ...

GD-EVT.
    evaluate event-type
    when msg-load-on-demand
         evaluate event-data-1
         when grlod-arrow-down-key
         when grlod-tab-key
              perform LOAD-FEW-RECORDS
         when grlod-page-down-key
         when grlod-scroll-bar-drag
              perform LOAD-SOME-RECORDS
         when grlod-ctrl-end-key
              perform LOAD-ALL-RECORDS
              modify gd cursor-y w-last-row
              set event-action to event-action-complete
         end-evaluate
    ...
```

performs different actions based on the key that the user pressed. In this example, scrolling using the arrow keys or using the mouse will load a set number of records, while pressing Ctrl+End will load all the records.

Date-entry

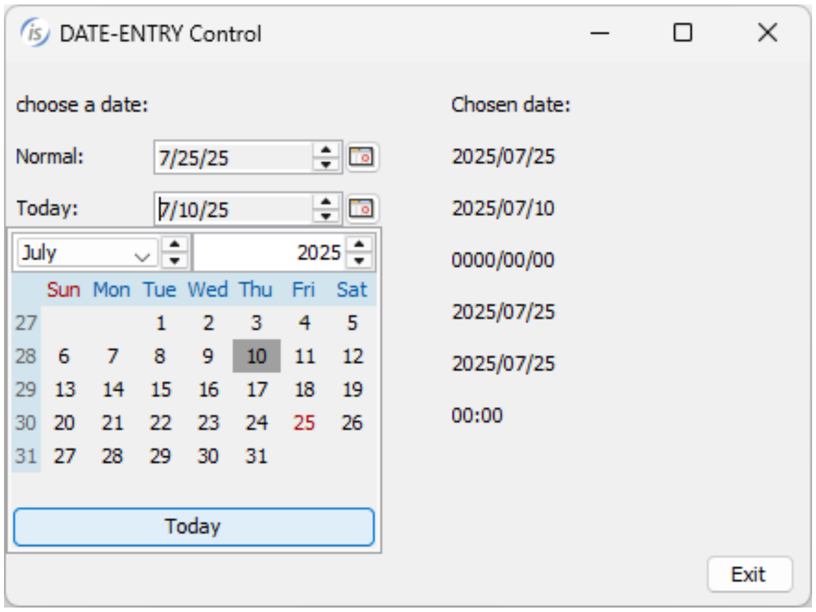
In the date-entry control, a new style named TODAY-BUTTON-VISIBLE has been implemented to show the “Today” button in the calendar opened to choose the date.

For example, the following code snippet:

```
03 date-entry today-button-visible
   line 12 col 14 size 20 cells
   ...
```

uses the new style, and running the program is shown in Figure 7, *Date-entry Today button*. When the “Today” button is pressed, the calendar closes, setting the value to the current date.

Figure 7. Date-entry Today button



choose a date:
Chosen date:

Normal: 7/25/25 — 2025/07/25
Today: 7/10/25 — 2025/07/10

0000/00/00
2025/07/25
2025/07/25
00:00

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
27			1	2	3	4	5
28	6	7	8	9	10	11	12
29	13	14	15	16	17	18	19
30	20	21	22	23	24	25	26
31	27	28	29	30	31		

July 2025

IsCOBOL WebClient

IsCOBOL WebClient 2025 R2 features upgraded components, such as Jetty embedded server, and supports Java 21. This version also supports idle instances to reduce the startup time and includes an embedded Session Pool in the Cluster Server.

<u>Upgraded components</u>

Jetty 12 is the integrated servlet container used to serve contents, and it supports the execution of WebClient with Java 21. This completes support of the latest Java LTS version for all Veryant products.

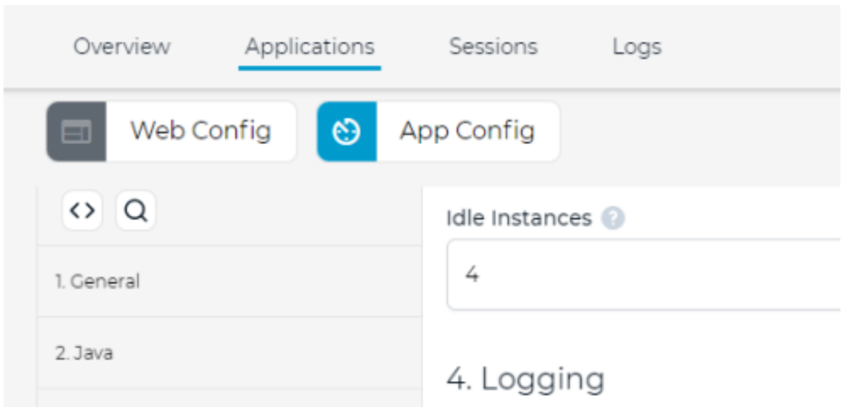
Jetty 12 requires at least Java 17 to run, and WebClient will now automatically fallback to using Tomcat as default servlet container when running with Java 1.8 or Java 11.

WebClient will handle the switch automatically and transparently.

<u>Idle instances</u>

This feature lets you define a number of instances of a program that will be automatically started before any user actually makes a request. The purpose of idle instances is to reduce the startup time users must wait by eliminating the JVM startup delay. You can configure how many idle instances should always be available in the "App config" section of the Admin Console. In Figure 8, *App configuration with 4 idle instances*, shows a configuration that sets 4 idle instances for the selected application.

Figure 8. App configuration with 4 idle instances.



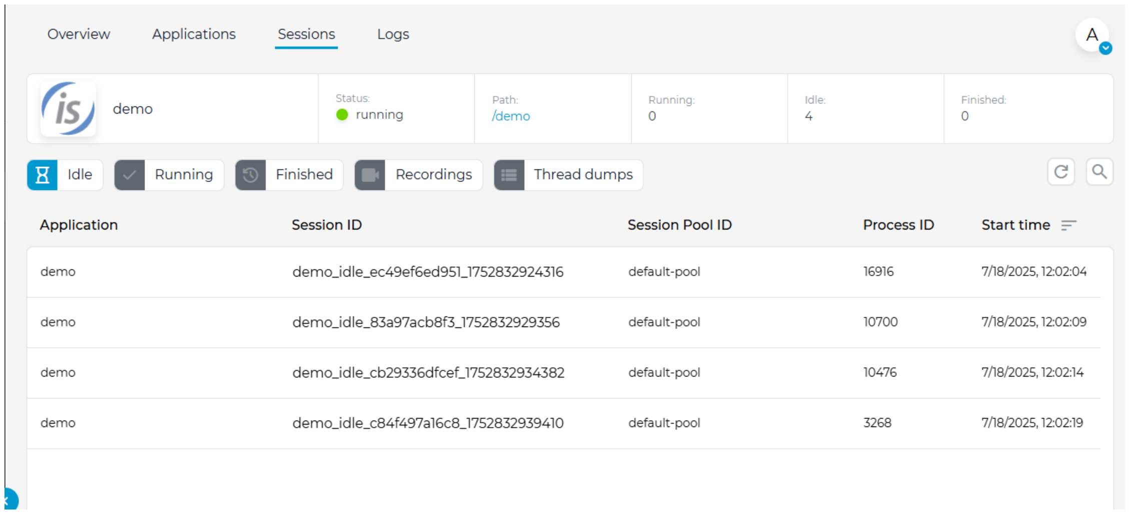
Idle Instances: 4

Using the above configuration, you will always find 4 additional Java processes in your system. These processes are created as soon as you save the modification to the configuration, and every time you restart the WebClient service.

Whenever a user requests the execution of the webapp, one of these idle Java processes will be used. Since it has already started, the user doesn't have to wait for the JVM initialization and will experience a faster application startup time. After a sleeping Java process has been used, the WebClient service will immediately start a new one so that the configured number of idle instances is always guaranteed.

You can see the count and the list of currently running idle instances in the Sessions view in the Admin Console, as depicted in Figure 9, *IDLE instances in the Sessions view*.

Figure 9. IDLE instances in the Sessions view.



Application	Session ID	Session Pool ID	Process ID	Start time
demo	demo_idle_ec49ef6ed951_1752832924316	default-pool	16916	7/18/2025, 12:02:04
demo	demo_idle_83a97acb8f3_1752832929356	default-pool	10700	7/18/2025, 12:02:09
demo	demo_idle_cb29336dfcef_1752832934382	default-pool	10476	7/18/2025, 12:02:14
demo	demo_idle_c84f497a16c8_1752832939410	default-pool	3268	7/18/2025, 12:02:19

When you change your app configuration, idle instances are automatically recreated.

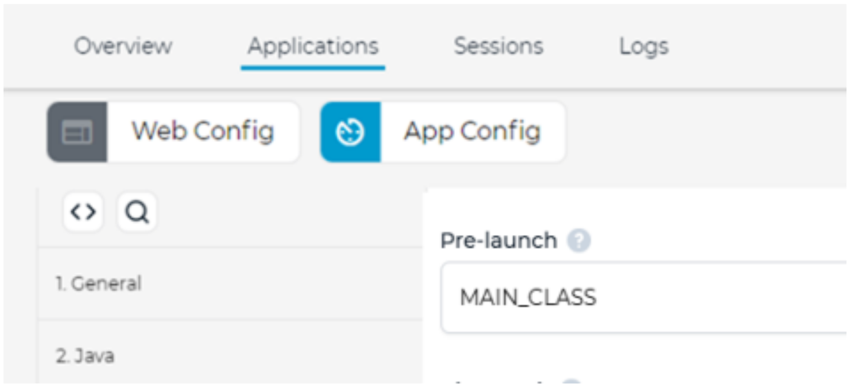
The Pre-launch configuration entry lets you choose if the application main class must be executed along with the idle JVM startup or only when the user requests a new session. The possible values are:

• NONE: The launch of idle instance stops right before calling the main class defined in the app launch configuration. The main class of your app will be called after the user requests a new instance. This is the default.

• MAIN_CLASS: This option launches the idle instance fully, including calling the main class defined in app launch configuration. When a user requests a new connection, the program will be instantly ready for use. This choice produces a faster startup but has some limitations: because the idle instance is launched before the user requests the program, user-defined variables in configuration parameters – such as userDir, vmArgs, etc. -- won't be recognized.

The next picture, Figure 10, *Pre-launch setting*, shows the Pre-launch option set to MAIN_CLASS.

Figure 10. Pre-launch setting.



Pre-launch: MAIN_CLASS

Embedded Session Pool in Cluster Server

The Cluster Server can now be started with a Session Pool included in the same java process. The embedded Session Pool is exactly the same as a standalone Session Pool. To start a Cluster Server with embedded Session Pool, use the `-clustersessionpool` command line option.

When you start the Cluster Server as a foreground process, pass the option on the webcclient-cluster command line, for example:

```
webcclient-cluster -clustersessionpool
```

When you start the Cluster Server as a Windows Service, pass the option on the sc command line, for example:

```
sc create "WebClientClusterEmbeddedSp" start= auto binPath=
"C:\Veryant\isCOBOL_WEBC2025R2\bin\webclient-cluster.exe -p 8080
-clustersessionpool
```

When you start the Cluster Server as a Unix daemon, pass these options through the WEBCLIENT_CL_OPTS environment variable, for example:

```
export WEBCLIENT_CL_OPTS=-p 8080 -clustersessionpool
webclient-cluster start
```

The embedded Session Pool will look for configuration files in the root directory, but you can also provide the path to the files through these command line options:

`-pfsp` /path/to/webclient-sessionpool.properties

`-csp` /path/to/webclient-app.config

To start the embedded Session Pool when deploying to Tomcat you can define these additional system properties:

```
webclient.clusterWithSessionPool=true
# custom configuration (optional)
webclient.propertiesFile.sessionPool=/path/to/webclient-sessionpool.properties
webclient.configFile.sessionPool=/path/to/webclient-app.config
```

Pinch gesture support

When running on mobile devices like smartphones and tablets, or more generally speaking when the UI is displayed on a touch screen, the user can pinch to zoom in and zoom out. This gesture was not supported in the previous WebClient versions, and pinching had no effect.

Compatibility improvements

isCOBOL 2025 R2 has been enhanced to improve compatibility with other COBOL dialects, such as MicroFocus COBOL and IBM COBOL. The compiler has been improved with added functions, custom functions can now be declared using the FUNCTION-ID syntax, and the CURRENCY symbols have been improved.

Functions

A typical COBOL program source code includes calling intrinsic functions that are part of the isCOBOL library. In this new release, four new functions have been implemented to provide better compatibility with other COBOL dialects.

Two of these handle the conversion from characters to bits and vice versa. These functions are BIT-OF and BIT-TO-CHAR.

Two handle the conversion from characters to hexadecimal values: HEX-OF and HEX-TO-CHAR

These functions require one alphanumeric parameter. The following is a code snippet that demonstrates how to use the new routines.

```
WORKING-STORAGE SECTION.
77  w-char pic x.
77  w-hex  pic x(2).
77  w-8bit pic x(8).
PROCEDURE DIVISION.
...
  move "A" to w-char
  move function bit-of(w-char) to w-8bit
  display w-8bit
  move function bit-to-char(w-8bit) to w-char
  display w-char
  display function hex-of(w-char)
  move "42" to w-hex
  display function hex-to-char(w-hex)
```

Custom functions, also known as user-defined functions, are supported in this new release, allowing developers to write their own function definitions that can then be invoked using the same FUNCTION keyword.

A new function must be declared using FUNCTION-ID paragraph, then it can be accessed from any other source code by declaring it in the repository.

For example, the following code declares and implements a new function using the function-id paragraph:

```
function-id. myfunction as "myfunction".
working-storage section.
77  myresult pic x(9).
linkage section.
77  myparam  pic x(5).
procedure division using myparam
               returning myresult.
    ...
    string myparam delimited by size
           "-END"  delimited by size
              into myresult
    goback.
    end function myfunction
```

And the code below shows how to declare and use the function from another COBOL source.

```
configuration section.
repository.
    function myfunction as "myfunction"
    ...
working-storage section.
77  w-par  pic x(5).
77  w-res  pic x(9).
procedure division.
    ...
    move function myfunction(my-par) to w-res
    display $myfunction(mypar)
```

The returning item of the user-defined function, in this example, is a string that is created by appending the string argument received as parameter and the suffix "-END". In the second source file, the function is invoked in the move statement, and the resulting string is used in the display statement.

CURRENCY symbols

In COBOL, the CURRENCY SIGN in the SPECIAL-NAMES is used in the PICTURE clause to represent a monetary value. The new isCOBOL release supports the declaration of multiple currency signs and adds support for the WITH PICTURE SYMBOL clause that is used in the item definition to provide the correct currency.

For example, the following code snippet declares three different currency signs:

```
configuration section.
special-names.
    currency sign is "USD " with picture symbol "$"
    currency sign is "EUR " with picture symbol "€"
    currency sign is "CHF " with picture symbol "F"
    ...
working-storage section.
01  w-price       pic 9(4)v99.
01  w-usd-price   pic $$,$$$.99.
01  w-eur-price   pic €€,€€€.99.
01  w-chf-price   pic FF,FFF.99.
procedure division.
...
    move 1234.89 to w-price
    move w-price to w-usd-price, w-eur-price, w-chf-price
    display w-usd-price
    display w-eur-price
    display w-chf-price
```

The output of the program in execution is:

```
USD 1,234.89
EUR 1,234.89
CHF 1,234.89
```

Runtime enhancements

The isCOBOL runtime has been enhanced with the ability to redirect CALL statements to a different program, along with new configuration options.

Redirect CALL

The existing configuration `iscobol.call_cancel.hook=classname` can now be used to redirect a CALL to another program and modify the parameters.

The com.iscobol.rts.CallHandler interface has been improved with the ability to redirect the call to another program by raising the dedicated exception com.iscobol.rts.MapCallException. For example, consider a program that calls another program named PROGA, passing a string and a number as parameters, e.g.

```
77 parx   pic x(5) value 'abc'.
77 par9   pic 9(3) value 0.
...
    call "PROGA" using parx, par9.
```

Using the configuration `iscobol.call_cancel.hook=MYCALLHANDLER`, the MYCALLHANDLER class-id program is invoked just before executing PROGA. The logic in the BeforeCall method of MYCALLHANDLER modifies the parameter values by turning the string uppercase and the number incremented by 1, then it redirects the call to a program named PROGB by raising a MapCallException, as demonstrated in the following code snippet:

```
class-id. MYCALLHANDLER as "MYCALLHANDLER" implements jCallhandler.
configuration section.
repository.
    class jCallHandler     as "com.iscobol.rts.CallHandler"
    class jCallOverExc     as "com.iscobol.rts.CallOverflowException"
    class mapcallexception as "com.iscobol.rts.MapCallException"
    class picx             as "com.iscobol.types.PicX"
    class pic9             as "com.iscobol.types.NumericVar"
    ...
method-id. mybeforeCall as "beforeCall" override.
working-storage section.
77  class-name    pic x any length.
77  parx          object reference picx.
77  par9          object reference pic9.
    ...
linkage section.
77  iscobol-call object reference jIscobolCall.
77  j-objects    object reference JObjects.
procedure division using iscobol-call, j-objects raising jCallOverExc.
   ...
    set class-name to self:>getName(iscobol-call)
    if class-name = "PROGA"
       set parx to j-objects(i) as picx
       move function upper-case(parx) to parx
       set par9 to j-objects(i) as pic9
       add 1 to par9
       raise mapcallexception:>new("PROGB", j-objects)
       ...
```

When the program is run, PROGB is called instead of PROGA, and it receives the modified parameter values. This feature may be useful in testing environments to simulate alternative scenarios without changing the COBOL programs code.

New configurations:

The following is a list of new properties introduced in this release, in both the Runtime and the utilities.

When parsing JSON strings, a new property named **`iscobol.json_parse.read_on_mismatch`** can be set to true to ignore errors when the JSON stream doesn't exactly match the COBOL group structure. The runtime will set the values of all COBOL items that have a matching item in the JSON stream, leaving the other items unchanged.

For COBOL programs running in character mode, **`iscobol.dispsign`** can be set to false to avoid showing the separated sign during character display.

When working with Java-beans controls in graphical interface programs running in ThinClient or WebClient environments, Java-bean methods should be serializable as they run on the client and need to be streamed from the server. When testing programs in standalone environments, where serialization is not mandatory, developers can now set the option **`iscobol.gui.javabean.serialization_check`** to true to force java-bean serialization. This setting helps developers test code as if it were running using the isCOBOL Application Server.

When working with databases and accessing data via JDBC drivers, the transaction isolation level can be set by using the configuration option **`iscobol.jdbc.transaction_isolation`**.

When using Database Bridge with EDBI routines for MSSQL, the setting `iscobol.jdbc.cursor.concurrency` can be set to `1007` for improved performance.

When accessing DBMaker with the native DCI interface, you can now set the configuration **`iscobol.dci_routines.use_connector`** to true to make DCI routines use the DCIC connector.

When handling errors, the configuration property **`iscobol.file.errors_ok`** can now be set to 3 to enable MF-compatible I/O errors.

In the ISUPDATER utility, a new entry named **`lib.iscobolJars`** can be defined in the configuration file to manage the SDK jars folder, useful when adding third party .jar files that are needed by the COBOL application.

IsCOBOL Server enhancements

The isCOBOL Server has been improved, allowing ThinClient sessions to have a custom timeout value for connection errors. TurboRun connections have been improved with the same capability.

ThinClient -timeout option

Usually, connection errors such as network connectivity issues or not having the ISCSERVER program running, are reported after a timeout period that is OS dependent. Now, the command line option `-timeout` can be used to force the number of seconds before errors are reported. If the default timeout is too high, the setting can be used to report such errors sooner.

In the following example, the isclient command uses 2 seconds as the timeout value for connections:

```
isclient -hostname myserver -port 10999 -timeout 2 PROGNAME
```

When the ThinClient is executed using an .istc file, the following new configuration needs to be set:

```
iscobol.timeout=2
```

Trun -timeout option

The same option is supported by the Turbo Run (trun) tool. For example:

```
trun -hostname myserver -port 10995 -timeout 2 PROGNAME
```

or using the configuration TRUN_TIMEOUT when executing locally:

```
export TRUN_TIMEOUT=2
trun PROGNAME
```

Debugger enhancements

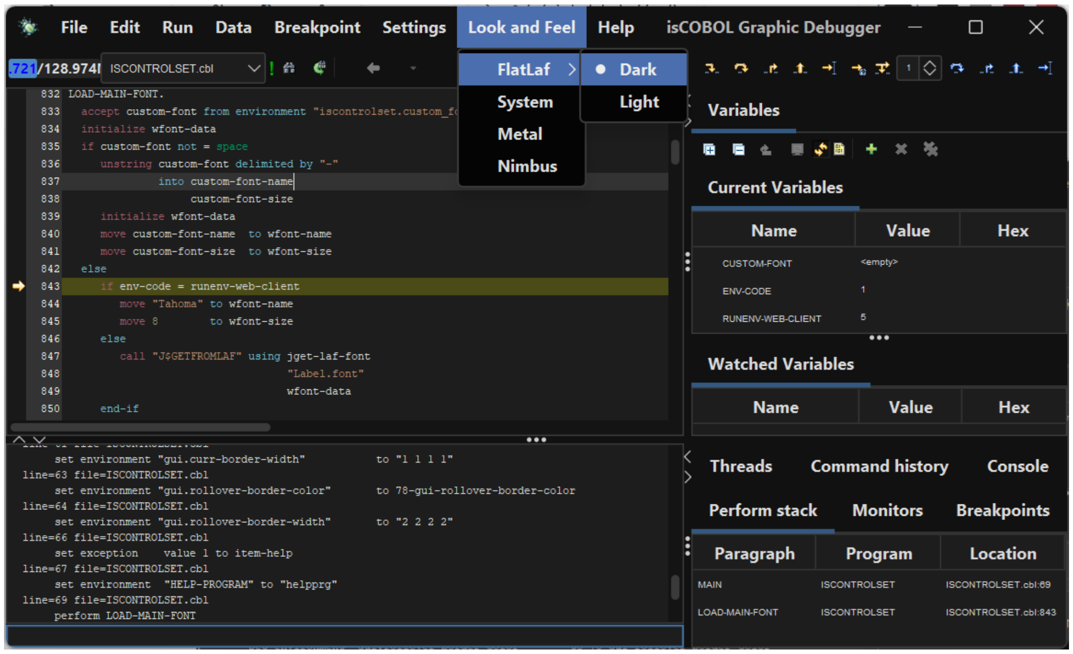
The isCOBOL Debugger has been improved, adding support for dark mode using FlatLAF and for changing the LAF on the fly.

Dark mode support

Developers frequently choose dark mode for coding due to reduced eye strain, improved focus, and energy conservation. The darker background minimizes glare, especially on glossy screens, and can be more comfortable for prolonged coding sessions. Dark mode also offers a visually appealing aesthetic and can improve battery life on devices with OLED or AMOLED displays. Developers have been able to activate the dark mode in development environments such as Eclipse and Visual Studio Code. Starting from this release, dark mode can also be activated in the COBOL Debugger, another great tool where developers usually spend a lot of their time.

The LAF (Look-And-Feel) in the Debugger can be changed on demand with the new menu item "Look and Feel", as shown in Figure 11, *Debugger dark mode*.

Figure 11. Debugger dark mode.



```
LOAD-MAIN-FONT.
  accept custom-font from environment "iscontrolset.custom_fc
  initialize wfont-data
  if custom-font not = space
     unstring custom-font delimited by "-"
              into custom-font-name
                   custom-font-size
     initialize wfont-data
     move custom-font-name  to wfont-name
     move custom-font-size  to wfont-size
  else
     if env-code = runenv-web-client
       move "Tahoma" to wfont-name
       move 8        to wfont-size
     else
       call "J$GETFROMLAF" using jget-laf-font
                                 "Label.font"
                                 wfont-data
     end-if
```

```
     set environment "gui.curr-border-width"          to "1 1 1 1"
line=63 file=ISCONTROLSET.cbl
     set environment "gui.rollover-border-color"      to 78-gui-rollover-border-color
line=64 file=ISCONTROLSET.cbl
     set environment "gui.rollover-border-width"      to "2 2 2 2"
line=66 file=ISCONTROLSET.cbl
     set exception    value 1 to item-help
line=67 file=ISCONTROLSET.cbl
     set environment  "HELP-PROGRAM" to "helpprg"
line=69 file=ISCONTROLSET.cbl
     perform LOAD-MAIN-FONT
```

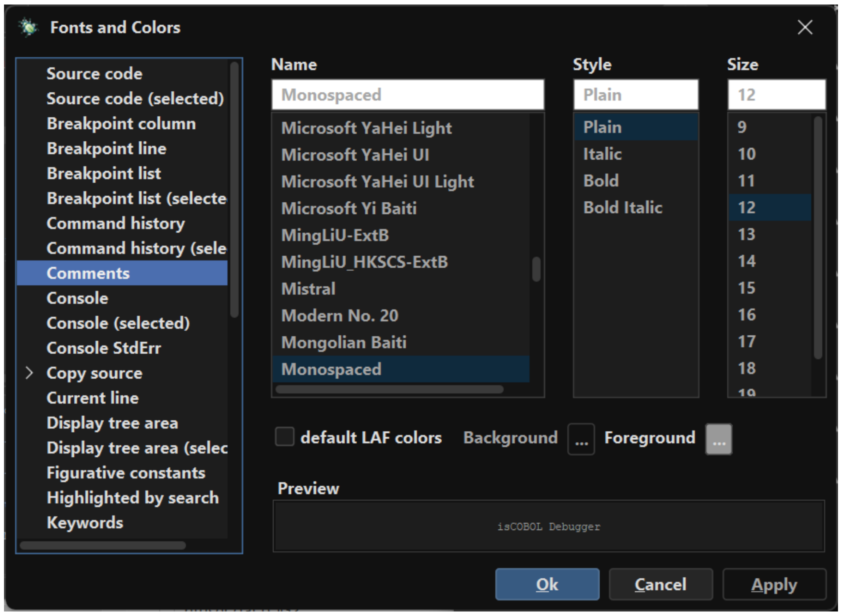
Name	Value	Hex
CUSTOM-FONT	<empty>	
ENV-CODE	1	
RUNENV-WEB-CLIENT	5	

Name	Value	Hex

Paragraph	Program	Location
MAIN	ISCONTROLSET	ISCONTROLSET.cbl:69
LOAD-MAIN-FONT	ISCONTROLSET	ISCONTROLSET.cbl:843

The “Fonts and Colors” settings now have two different sets of colors: one is the equivalent of the previous light theme, and one for the new dark theme. The color settings are saved in separate sections of the configuration for light and dark themes, allowing for full customization. Figure 12, *Debugger color settings,* a gray color is used for COBOL comments.

Figure 12. Debugger color settings.



IsCOBOL IDE enhancements

isCOBOL IDE, the Integrated Development Environment based on Eclipse, supports importing projects and settings. The new release adds support for new types of imports from a command line and improves usability in screen and report painters.

Import from command line

isCOBOL's IDE implements several command-line switches that allow you to perform tasks in the background without user interaction. This is useful to automate the creation of a workspace using components that are needed. The new release adds new import capabilities for:

- isCOBOL Data Layout files (.idl extension),
- isCOBOL Screen Program files (.isp extension),
- isCOBOL Wow Program files (.wsp extension),
- R/M Wow Program files (.wpj extension)

For example, to import all the .idl files saved in \myproject\idl and all the .isp files saved in \myproject\isp to an empty workspace, these 2 commands can be executed:

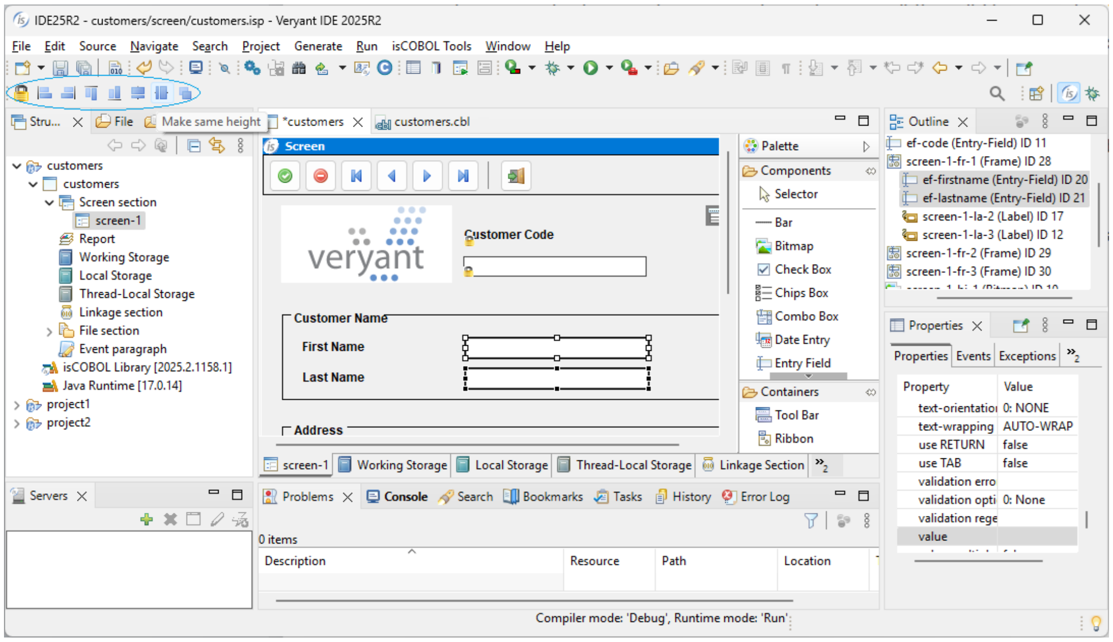
```
isIDE -data \new-workspace -nosplash --launcher.suppressErrors –application
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importIdlApplication
project newproject folder \myproject\idl

isIDE -data \New-workspace -nosplash --launcher.suppressErrors –application
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importIspApplication
project newproject folder \myproject\isp
```

Painters’ usability

A new toolbar with buttons for alignment and lock features has been added to the IDE for screen and report painters. These buttons improve the usability for developers that prefer working with a mouse, as shown in Figure 13, *Painter’s new buttons*.

Figure 13. Painter’s new buttons.



Property	Value
text-orientatioı	0: NONE
text-wrapping	AUTO-WRAP
use RETURN	false
use TAB	false
validation erro	
validation opti	0: None
validation rege	
value	

Additionally, a new setting "Generate copy books in the 'Screen Program Copy' folder" has been added in “WOW - Code Generator” to customize the folder where .prd and .wrk copy files are generated.

Visual Studio Code enhancements

The Veryant extension for Visual Studio Code has been enhanced and now has workspace support, debug/release compilation mode, and the ability to debug remote processes.

Workspace support

Visual Studio Code supports grouping folders and projects in “workspaces”. Starting from this release, Veryant Extension for Visual Studio Code supports workspaces, and each project folder can have its own settings for both compiler and runtime options.

Supporting workspaces allows developers to open multiple projects, compile and run each using project-specific SDK, JDK and JRE versions, compiler options such as compiler switches and compiler libraries, and run programs with specific runtime versions and options, such as configuration settings and runtime libraries. The “Compile Project(s)” command has been enhanced to compile all sources for every open project. When selecting sources belonging to different projects and invoking the “Compile selected files”, the command will apply the correct project compiler options.

When working with workspaces, settings can be saved at the project (folder) level, at workspace level, or at user level. Settings are stored in the relative folder and are applied with the correct priority: project setting, workspace settings, user settings.

Debug/release compilation

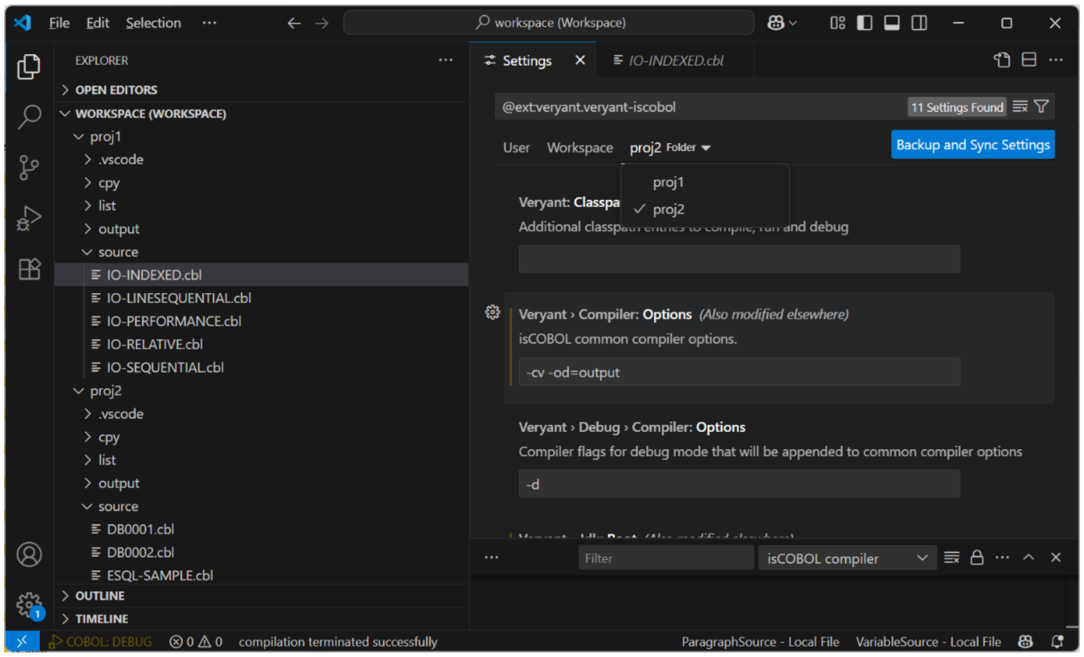
A new compilation switch has been added to globally set the compile mode to either Debug or Release. When compiling sources, the compiler options are set by combining the content of the veryant.compiler.options variable with the compile-mode specific options. The mode-specific compiler options are stored in the settings "veryant.debug.compiler.option” and "veryant.release.compiler.options".

The active mode is displayed in the lower-left portion of the status bar and clicking it will invoke the switch selector in the “command section” of the editor, where a new mode can be selected.

Typical settings that are used when compiling in debug mode include the –d or –dx flags. Additional settings include the –od flag to specify different output folders for release-mode and debug-mode classes.

In Figure 14, *VS Code settings*, shows settings for compiler options flags set for a specific folder of a workspace.

Figure 14. VS Code settings.



@ext:veryant.veryant-iscobol

Veryant: Classpa
Additional classpa... and debug

Veryant › Compiler: **Options** *(Also modified elsewhere)*
isCOBOL common compiler options.
-cv -od=output

Veryant › Debug › Compiler: **Options**
Compiler flags for debug mode that will be appended to common compiler options
-d

⊗ 0 ⚠ 0 compilation terminated successfully

Remote debugging

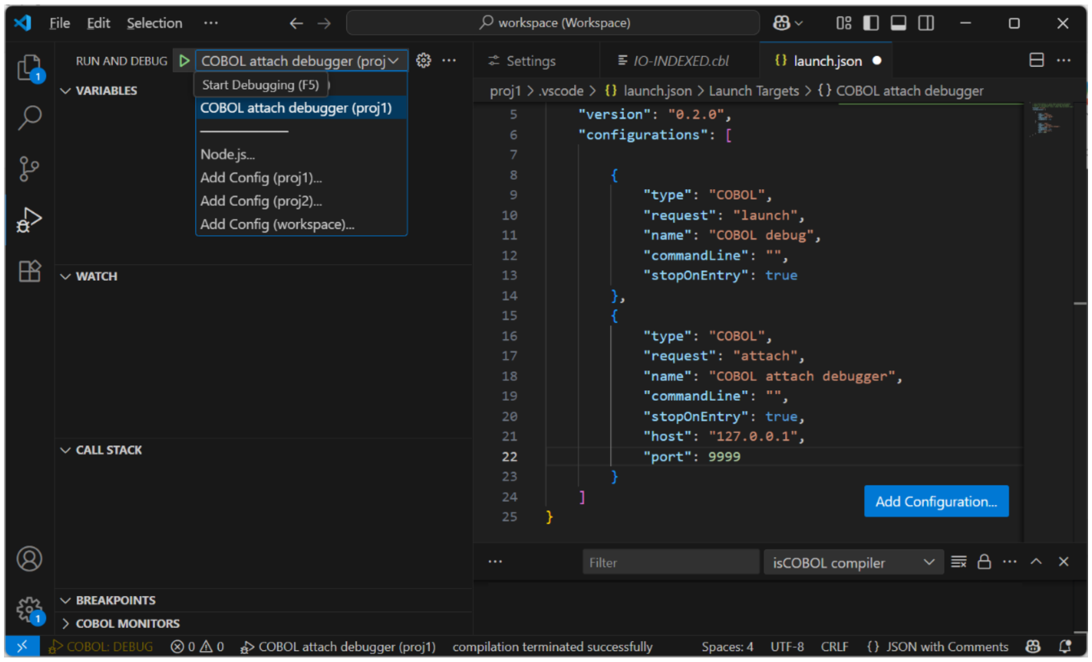
The new release of the Visual Studio Code extension can now be used to attach a debugging session to a remote program by specifying the IP/hostname and port number in the launch configuration settings.

This can be useful to debug processes running on a remote server or when the code is run outside Visual Studio Code; for example when the COBOL program is started from a C or Java process.

Use the runtime option iscobol.rundebug=1 or 2 to instruct the VS Code extension to attach the debugging session to a running process.

In Figure 15, *VS Code attach debugger*, we see the new "attach" request type for the debugger that allows remote debugging of processes running outside Visual Studio Code.

Figure 15. Visual Studio Code attach debugger.



```
    "version": "0.2.0",
    "configurations": [

        {
            "type": "COBOL",
            "request": "launch",
            "name": "COBOL debug",
            "commandLine": "",
            "stopOnEntry": true
        },
        {
            "type": "COBOL",
            "request": "attach",
            "name": "COBOL attach debugger",
            "commandLine": "",
            "stopOnEntry": true,
            "host": "127.0.0.1",
            "port": 9999
        }
    ]
}
```

IsCOBOL Utilities

IsCOBOL 2025 R2 has an enhanced GIFE utility that can filter records in multi table files, and the Profiler utility can now include or exclude specific programs or paragraphs from profiling.

GIFE

The GIFE utility now provides improved support for conditions with TABLENAME syntax in the EFD WHEN directive. The GIFE toolbar provides a button that lists the table names. If the user selects an item from the list, the grid in the List view is filtered with the records that satisfy the conditions for that table. There is also a check-box named “Filter Recs” that can be toggled to activate the filter when navigating records in the Byte view or Field view.

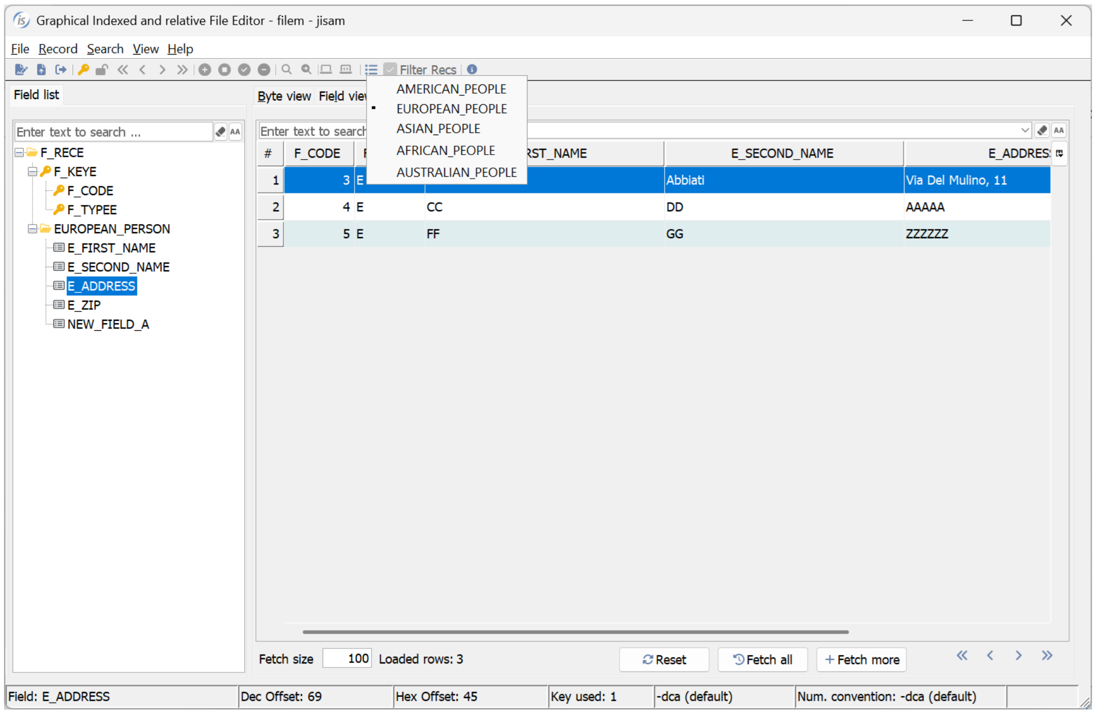
For example, in the following file description:

```
fd  filem.
>>EFD WHEN F_TYPE = "M" TABLENAME = AMERICAN_PEOPLE
01  f-recM.
    ...
>>EFD WHEN F_TYPE = "E" TABLENAME = EUROPEAN_PEOPLE
01  f-recE.
    ...
>>EFD WHEN F_TYPE = "S" TABLENAME = ASIAN_PEOPLE
01  f-recS.
    ...
>>EFD WHEN F_TYPE = "F" TABLENAME = AFRICAN_PEOPLE
01  f-recF.
    ...
>>EFD WHEN F_TYPE = "U" TABLENAME = AUSTRALIAN_PEOPLE
01  f-recU.
    ...
```

the EFD WHEN directives are used to specify 5 different table names, and GIFE uses this information to filter records belonging to the specific table name.

Figure 16, *GIFE table names list*, shows the record filter feature in use.

Figure 16. VS GIFE table names list.



#	F_CODE	F	RST_NAME	E_SECOND_NAME	E_ADDRES
1	3	E		Abbiati	Via Del Mulino, 11
2	4	E	CC	DD	AAAAA
3	5	E	FF	GG	ZZZZZZ

Fetch size 100 Loaded rows: 3

Field: E_ADDRESS | Dec Offset: 69 | Hex Offset: 45 | Key used: 1 | -dca (default) | Num. convention: -dca (default)

In addition, the GIFE utility now supports command line options to supply settings for the iscobol.file.prefix and iscobol.gife.efd_directory settings.

For example, the following command:

```
iscrun -c gife.properties -utility GIFE filem filem.xml
```

opens the file named filem using the dictionary file filem.xml using paths found in the configuration settings:

```
iscobol.gife.efd_directory=/myapp/efd
iscobol.file.prefix=/myapp/custom_data:/myapp/common_data
...
```

Profiler

The Profiler output might be rather complex as all the paragraphs of the profiled programs are profiled. In previous releases, settings allowed you to specify the programs to include or exclude in the profiling. Since version 2025 R2 it's possible to reduce the list of profiled paragraphs using the includes and excludes settings and the syntax "::" in the iscobol.profiler.includes and iscobol.profiler.excludes configurations options.

For example of specifying a program to include, the following command executes the IO-PERFORMANCE sample by profiling only the IO-INDEXED subprogram; the other programs called by IO-PERFORMANCE (IO-LINESEQUENTIAL, IO-RELATIVE and IO-SEQUENTIAL) are not included in the profiler report:

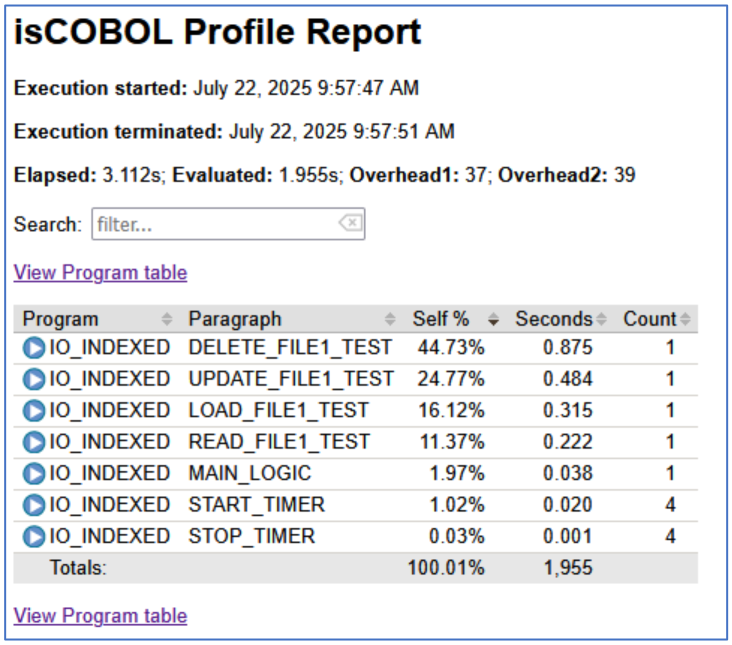
```
iscrun -c prof.properties -profile IO_PERFORMANCE
```

where the configuration file contains:

```
iscobol.profiler.includes=IO_INDEXED
```

The above command produces a report as shown in Figure 17, *Profile all paragraphs*.

Figure 17. Profile all paragraphs.



isCOBOL Profile Report

Execution started: July 22, 2025 9:57:47 AM

Execution terminated: July 22, 2025 9:57:51 AM

Elapsed: 3.112s; **Evaluated:** 1.955s; **Overhead1:** 37; **Overhead2:** 39

Search: filter...

View Program table

Program	Paragraph	Self %	Seconds	Count
IO_INDEXED	DELETE_FILE1_TEST	44.73%	0.875	1
IO_INDEXED	UPDATE_FILE1_TEST	24.77%	0.484	1
IO_INDEXED	LOAD_FILE1_TEST	16.12%	0.315	1
IO_INDEXED	READ_FILE1_TEST	11.37%	0.222	1
IO_INDEXED	MAIN_LOGIC	1.97%	0.038	1
IO_INDEXED	START_TIMER	1.02%	0.020	4
IO_INDEXED	STOP_TIMER	0.03%	0.001	4
Totals:		100.01%	1,955	

View Program table

Expanding on this example, the following command extends the previous one by profiling only the READ-FILE1-TEST paragraph and the UPDATE-FILE1-TEST paragraph of the IO-INDEXED program; the other paragraphs of IO-INDEXED are not included in the profiler:

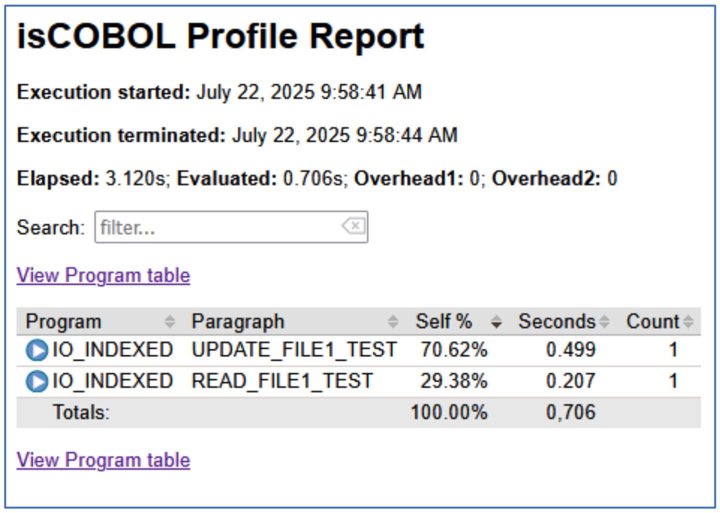
```
iscrun -c prof.properties -profile IO_PERFORMANCE
```

where the configuration file contains:

```
iscobol.profiler.includes=IO_INDEXED::READ_FILE1_TEST,IO_INDEXED::UP
DATE_FILE1_TEST
```

The above command produces a report as shown in Figure 18, *Profile specific paragraphs*.

Figure 18. Profile some paragraphs.



isCOBOL Profile Report

Execution started: July 22, 2025 9:58:41 AM

Execution terminated: July 22, 2025 9:58:44 AM

Elapsed: 3.120s; **Evaluated:** 0.706s; **Overhead1:** 0; **Overhead2:** 0

Search: filter...

View Program table

Program	Paragraph	Self %	Seconds	Count
IO_INDEXED	UPDATE_FILE1_TEST	70.62%	0.499	1
IO_INDEXED	READ_FILE1_TEST	29.38%	0.207	1
Totals:		100.00%	0,706	

View Program table

The reported elapsed time is the same as the previous report, but the evaluated time is lower, as it only includes specific paragraphs.