veryant

isCOBOL^{TM} Evolve

isCOBOL Evolve 2022 Release 2 Overview

isCOBOL Evolve 2022 Release 2 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2022 R2.

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

The new compiler supports an integrated PreProcessing feature and increased compatibility and performances with other COBOL dialects.

The new WebClient release now supports Java 17 and offers several improvements and higher performance on slow networks.

isCOBOL Evolve 2022 R2 GUIs are now Hi-DPI aware and include improvements, refinements, and new compatibility features to simplify the migration of existing COBOL programs to the isCOBOL Evolve suite.

Details on these enhancements and updates are included below.

isCOBOL Compiler

The isCOBOL 2022 R2 compiler includes a new feature that allows developers to write a PreProcessor in COBOL or in Java to integrate custom logic to be applied to source code. Compatibility with other COBOL dialects and performances has been improved to simplify migrations, and isCOBOL ESQL has been improved to ease migration from IBM DB2 PRE and Oracle Pro*COBOL (ESQL precompilers).

PreProcessor integration

A new compiler configuration,"iscobol.compiler.custompreproc", has been added to enable Preprocessor parsing. Set this configuration to one or more classes that implement the isCOBOL compiler interface "com.iscobol.compiler.custpreproc.LinePreProcessor". For example, when compiling a program with this set in the configuration:

```
iscobol.compiler.custompreproc=MyPreProc SecondPreProc
```

the compiler will pass every line of the source code being compiled to the MyPreProc class, which may perform source code changes. The resulting line is then passed to the SecondPreProc class, which can perform additional changes, and the resulting source code line is finally passed to the compiler for compilation.

isCOBOL already supports the use of the compiler option "`iscobol.compiler.regexp`" that can be used to replace text within the source code, but the new PreProcessor feature is more flexible and allows the parsing of multiple lines of code for analysis.

Both configuration options can be used simultaneously, letting you make complex code substitutions. The regexp will be executed first, then the custom preprocessor.

To implement the com.iscobol.compiler.custpreproc.LinePreProcessor interface, the class needs to include a "process" method with the signature:

```
public void process(java.lang.String originalLine,
                    int sourceFormat,
                    java.lang.String fileName,
                    int lineNumber,
                    java.lang.String[] compilerOptions,
                    com.iscobol.compiler.custpreproc.ProcessResult result)
             throws com.iscobol.compiler.custpreproc.ProcessException
```

So, the source can be written in COBOL declaring a CLASS-ID that has a METHOD-ID receiving these parameters:

```
class-id. MyPreProc as "MyPreProc" implements LinePreProcessor.
repository.
    class LinePreProcessor as "com.iscobol.compiler.custpreproc.LinePreProcessor"
    class ProcessResult    as "com.iscobol.compiler.custpreproc.ProcessResult"
    class ProcessException as "com.iscobol.compiler.custpreproc.ProcessException"
...
method-id. process as "process".
linkage section.
77  original-line     object reference "java.lang.String".
77  source-format     object reference "int".
77  source-file-name  object reference "java.lang.String".
77  line-number       object reference "int".
77  compiler-options  object reference "java.lang.String[]".
77  result            object reference ProcessResult.
procedure division using original-line,
                         source-format,
                         source-file-name,
                         line-number,
                         compiler-options,
                         result
                 raising ProcessException.
```

For example, this method can be used to implement custom logic to write comments for a block of code, or to replace some code with another. You can also return Warnings or Errors exactly like the standard compiler produces. Another example use of the PreProcessor is to enforce or suggest code changes to comply with a desired standard set of rules.

When compiling sources with the -lf option, the resulting .list file will contain the output after the processing executed by the preprocessor classes is finished.

When debugging a COBOL program compiled using the preprocessor, the source shown will be the original source, as written by the developer, but the instructions executed will be the result of the preprocessor output.

To better follow the logic of the class-id MyPreProc, you can debug the COBOL source with the Remote Debugger feature. To do that, first compile with this variable set in the configuration:

```
iscobol.rundebug=2
```

and start the Remote Debugger with the command:

```
iscrun -d -r
```

This will start the Remote Debugger on the default localhost using the default port number 9999 where the compilation process is waiting for the debugger execution. If the compilation is executed on a different server, you can pass the hostname of that server when launching the Remote Debugger.

For additional details on this feature, refer to the updated 2022 R2 documentation and new samples installed in the sample folder “compiler-pre-process”.

Option to generate optimized class

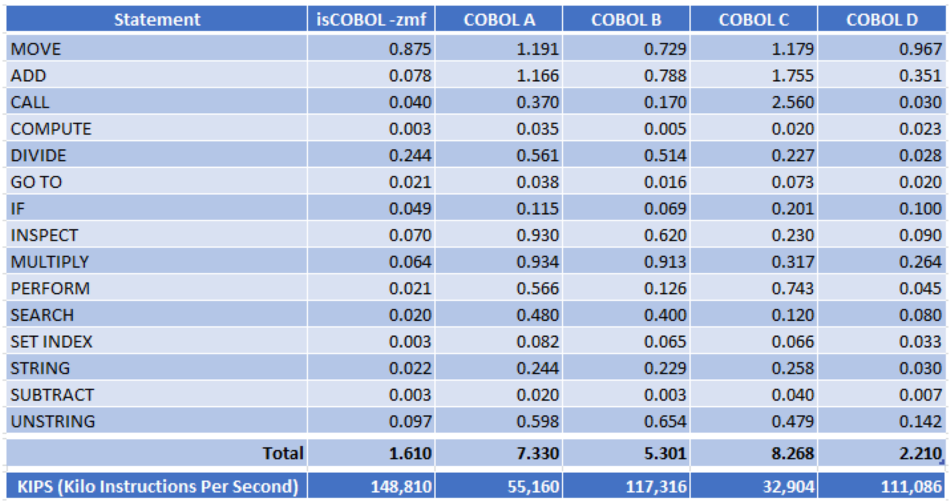
isCOBOL 2022 R2 provides a new compiler option named "-zmf" that internally activates a new compiler engine designed to maximize runtime performance and minimize resource usage. The new compiler engine has been developed for maximum compatibility with Enterprise IBM Cobol and Micro Focus COBOL. This early version can be used for batch COBOL programs without user interface. There are some syntax limitations as well in this first version.

The new compiler engine uses top-down (recursive descent) parsers as opposed to bottom-up parsers generated by a YACC-like tool. Based on the most popular parser generator, JavaCC, the new compiler engine runs in three steps:

- COBOL syntax validation
- Walk to collect data and to generate functions
- Emitters, LAMBDA based to emit Java classes

As shown Figure 1, *NCR CSAMPLE Benchmark*, performance comparison was made using NCR CSAMPLE benchmark test between isCOBOL 2022 R2 with "-zmf" and other COBOL compilers from the market. The test was run on Windows 11 Pro 64-bit with i7-8550U CPU @ 1.80GHz and 16 GB of RAM. The test shows that isCOBOL's new compiler engine beat our competitors by running 13% - 80% faster and executing at least 22% more instructions per second.

Figure 1. NCR CSAMPLE Benchmark.



Statement	isCOBOL -zmf	COBOL A	COBOL B	COBOL C	COBOL D
MOVE	0.875	1.191	0.729	1.179	0.967
ADD	0.078	1.166	0.788	1.755	0.351
CALL	0.040	0.370	0.170	2.560	0.030
COMPUTE	0.003	0.035	0.005	0.020	0.023
DIVIDE	0.244	0.561	0.514	0.227	0.028
GO TO	0.021	0.038	0.016	0.073	0.020
IF	0.049	0.115	0.069	0.201	0.100
INSPECT	0.070	0.930	0.620	0.230	0.090
MULTIPLY	0.064	0.934	0.913	0.317	0.264
PERFORM	0.021	0.566	0.126	0.743	0.045
SEARCH	0.020	0.480	0.400	0.120	0.080
SET INDEX	0.003	0.082	0.065	0.066	0.033
STRING	0.022	0.244	0.229	0.258	0.030
SUBTRACT	0.003	0.020	0.003	0.040	0.007
UNSTRING	0.097	0.598	0.654	0.479	0.142
Total	**1.610**	**7.330**	**5.301**	**8.268**	**2.210**
KIPS (Kilo Instructions Per Second)	**148,810**	**55,160**	**117,316**	**32,904**	**111,086**

Improved Compatibility with other COBOLs

Enhancements have been implemented in this release to improve compatibility with MicroFocus COBOL, such as:

- support for `THREAD-LOCAL-STORAGE SECTION` and `IS THREAD-LOCAL` clauses.

The THREAD-LOCAL-STORAGE Section describes data which is unique to each thread, useful for resolving contention problems related to concurrent access to the same data-item. If a data-item is declared inside this new SECTION, every thread has its own copy of the variable instead of sharing the same with other threads.

The IS THREAD-LOCAL clause can be used in the WORKING-STORAGE SECTION or on the FD level to declare that a data-item or File Record needs to be treated with the thread logic described above, making every thread access its separate variables or record definitions.

Here is a code snippet that shows how to declare variables in the THREAD-LOCAL-STORAGE SECTION and variables and FD with the THREAD-LOCAL clause:

```
file section.
fd  file-prod is thread-local.
01  prod-rec.
    03 prod-key  pic 9(4).
    03 prod-desc pic x(90).
working-storage section.
77  thr1         handle of thread.
77  thr2         handle of thread.
77  counter      pic 9(4) value 0 is thread-local.
thread-local-storage section.
01  w-rec.
    03 w-key     pic 9(4).
    03 w-desc    pic x(90).
```

- new compiler option -smfu to manage the Underscore in sources with MF compatibility

With isCOBOL, underscore and hyphen are treated as the same character, but in the MicroFocus compiler the underscore and hyphen are 2 different characters. When a COBOL program contains the same variable or paragraph name, with the only difference being an underscore and hyphen, the isCOBOL compiler will throw a Severe “Ambiguous identifier” and “Duplicate procedure name” error message when compiling.

With the new compiler option –smfu, isCOBOL treats underscore and hyphen as 2 different characters, allowing sources as described to be compiled without errors.

For example, the following code can be compiled correctly:

```
working-storage section.
77  my-var pic x.
77  my_var pic x.
...
procedure division.
...
    move "a" to my-var.
    move "b" to my_var.
    perform SHOW-MY-VAR.
    perform SHOW-MY_VAR.
SHOW-MY-VAR.
    display "my-var=" my-var.
SHOW-MY_VAR.
    display "my_var=" my_var.
```

- the semicolon is now accepted by the compiler as universal path separator

This simplifies sharing file paths between operating systems. Windows uses the ";" as a path separator, while MacOS and Linux use ":". The isCOBOL compiler will translate the universal path separator, ";", to the OS-specific character.

For example, the following command used to compile on Windows:

```
iscc -sp=copylib;other-copy/cpy program-name
```

can now be used without changes when compiling in Linux or MacOS.

This change also benefits the isCOBOL IDE when sharing the same Workspace to compile programs using different operating systems such as Windows, Linux and MacOS.

Enhancements have been made in this release to improve compatibility with IBM COBOL, such as:

- support for `JSON PARSE` and `JSON GENERATE` statements.

The existing XML PARSE and XML GENERATE statements gave you the ability to manage XML streams, and starting from this release, the `JSON PARSE` and `JSON GENERATE` statements are supported to manage JSON streams.

IsCOBOL already supports the “com.iscobol.rts.JSONStream” class to manage JSON streams, but now direct migration from IBM COBOL sources that contain these statements is supported by the compiler. Here’s a code snippet of the instructions:

```
 working-storage section.
...
01 Grp.
   05 Acc-No   pic AA9999.
   05 More.
      10 code  pic S99V9 occurs 2.
   05 cr-date  pic 99/99/9999.
01 json-stream pic x(80).
01 i           binary pic 99.
procedure division.
...
    initialize grp
    move "AB1234"     to acc-no
    move 1.2          to code(1)
    move -3           to code(2)
    move "01/07/2022" to cr-date
    JSON GENERATE json-stream from grp
      count i
      name of code is 'Value'
      suppress cr-date
    on exception
      display "Error on JSON generate"
    not on exception
      display "Successful JSON generate"
    end-json.

    initialize grp
    JSON PARSE json-stream into grp
      with detail
      name of code is 'Value'
    end-json.
```

The generated JSON will be:

```
{"Grp":{"Acc-No":"AB1234","More":{"Value":[1.2,-3.0]}}}
```

When the COBOL program receives a JSON stream, the JSON PARSE statement will parse the stream and set the values in the corresponding WORKING STORAGE items.

Improved Compatibility with other ESQL Precompilers

Enhancements have been made in this release to improve compatibility with RDBMS PreCompilers such as IBM DB2 PRE and Oracle Pro*COBOL.

- Support for Common Table Expressions (CTE)

A Common Table Expression (CTE) is the result set of a query which exists temporarily and for use only within the context of a larger query. The result of a CTE is not stored and exists only for the duration of the query. CTEs enable users to easily write and maintain complex queries with increased readability and simplification. This reduction in complexity is achieved by deconstructing ordinarily complex queries into simple blocks to be used and reused as necessary in rewriting the query. CTEs initiate with the WITH keyword. Here's a code snippet as an example:

```
EXEC SQL
     DECLARE CUR CURSOR FOR
     WITH Sales_CTE AS
( SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
         INTO :wrk-person-id, :wrk-order-id, :wrk-year
         FROM Sales.SalesOrderHeader
        WHERE SalesPersonID IS NOT NULL )

SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
       FROM Sales_CTE
       GROUP BY SalesYear, SalesPersonID
       ORDER BY SalesPersonID, SalesYear
END-EXEC.
```

Note that some databases might not support this syntax, so ensure it's supported by your RDBMS before using. The larger RDBMS like IBM DB2, Oracle, Postgres, and Microsoft SQL Server support it.

- Ability to specify stored procedure and function names through host variables

Previously, the ESQL CALL statement required the procedure or function name to be a constant string. Now in 2022 R2 the name can be specified using host variables. For example:

```
MOVE "myproc" TO WRK-PROC-NAME.
EXEC SQL
     CALL :wrk-proc-name (:wrk-param-1 IN, :wrk-param-2 IN)
          INTO :wrk-result
END-EXEC.
```

This feature lets you write more dynamic code.

- Support for Oracle Pro*COBOL SQLDA structure

The compatibility with ESQL preprocessors has been increased in this version by supporting the Oracle Pro*COBOL format of the SQLDA structure. An SQLDA (SQL Descriptor Area) is a set of group items that store all the information the database needs about select-list items or place-holders for bind variables, except their values.

Previously, SQLDA was supported only in compatibility with the IBM DB2 preprocessor. From the new isCOBOL 2022 R2 release, SQLDA is also supported with Oracle Pro*COBOL.

The SQLDA structure from Oracle Pro*COBOL is defined as follows:

```
       01  SELDSC.
           02  SQLDNUM                    PIC S9(9) COMP VALUE 100.
           02  SQLDFND                    PIC S9(9) COMP.
           02  SELDVAR                    OCCURS 100 TIMES.
               03  SELDV                  PIC S9(9) COMP.
               03  SELDFMT                PIC S9(9  COMP.
               03  SELDVLN                PIC S9(9) COMP.
               03  SELDFMTL               PIC S9(4) COMP.
               03  SELDVTYP               PIC S9(4) COMP.
               03  SELDI                  PIC S9(9) COMP.
               03  SELDH-VNAME            PIC S9(9) COMP.
               03  SELDH-MAX-VNAMEL       PIC S9(4) COMP.
               03  SELDH-CUR-VNAMEL       PIC S9(4) COMP.
               03  SELDI-VNAME            PIC S9(9)
COMP.
               03  SELDI-MAX-VNAMEL       PIC S9(4) COMP.
               03  SELDI-CUR-VNAMEL       PIC S9(4) COMP.
               03  SELDFCLP               PIC S9(9) COMP.
               03  SELDFCRCP              PIC S9(9) COMP.
       01  XSELDI.
```

```
          03  SEL-DI            OCCURS 100 TIMES PIC S9(4) COMP.
      01  XSELDIVNAME.
          03  SEL-DI-VNAME      OCCURS 100 TIMES PIC X(80).
      01  XSELDV.
          03  SEL-DV            OCCURS 100 TIMES PIC X(80).
      01  XSELDHVNAME.
          03  SEL-DH-VNAME      OCCURS 100 TIMES PIC X(80).

      01  BNDDSC.
          02  SQLDNUM                    PIC S9(9) COMP VALUE 100.
          02  SQLDFND                    PIC S9(9) COMP.
          02  BNDDVAR                    OCCURS 100 TIMES.
              03  BNDDV                  PIC S9(9) COMP.
              03  BNDDFMT                PIC S9(9) COMP.
              03  BNDDVLN                PIC S9(9) COMP.
              03  BNDDFMTL               PIC S9(4) COMP.
              03  BNDDVTYP               PIC S9(4) COMP.
              03  BNDDI                  PIC S9(9) COMP.
              03  BNDDH-VNAME            PIC S9(9) COMP.
              03  BNDDH-MAX-VNAMEL       PIC S9(4) COMP.
              03  BNDDH-CUR-VNAMEL       PIC S9(4) COMP.
              03  BNDDI-VNAME            PIC S9(9) COMP.
              03  BNDDI-MAX-VNAMEL       PIC S9(4) COMP.
              03  BNDDI-CUR-VNAMEL       PIC S9(4) COMP.
              03  BNDDFCLP               PIC S9(9) COMP.
              03  BNDDFCRCP              PIC S9(9) COMP.
      01  XBNDDI.
          03  BND-DI            OCCURS 100 TIMES PIC S9(4) COMP.
      01  XBNDDIVNAME.
          03  BND-DI-VNAME      OCCURS 100 TIMES PIC X(80).
      01  XBNDDV.
          03  BND-DV            OCCURS 100 TIMES PIC X(80).
      01  XBNDDHVNAME.
          03  BND-DH-VNAME      OCCURS 100 TIMES PIC X(80).
```

These data items are used in DESCRIBE, OPEN and FETCH ESQL statements, for example:

```
  EXEC SQL  DESCRIBE BIND VARIABLES FOR Stmt1 INTO BNDDSC END-EXEC.
  EXEC SQL  DESCRIBE SELECT LIST FOR Stmt1 INTO SELDSC  END-EXEC.
  EXEC SQL  OPEN Cur1 USING DESCRIPTOR BNDDSC END-EXEC.
  EXEC SQL  FETCH Cur1 USING DESCRIPTOR SELDSC END-EXEC.
```

GUI enhancements

Many improvements to GUI controls have been implemented in this release. The isCOBOL framework is now fully dpi-aware. New properties and events are supported on several controls to enhance and improve GUI applications.

Fully dpi-aware

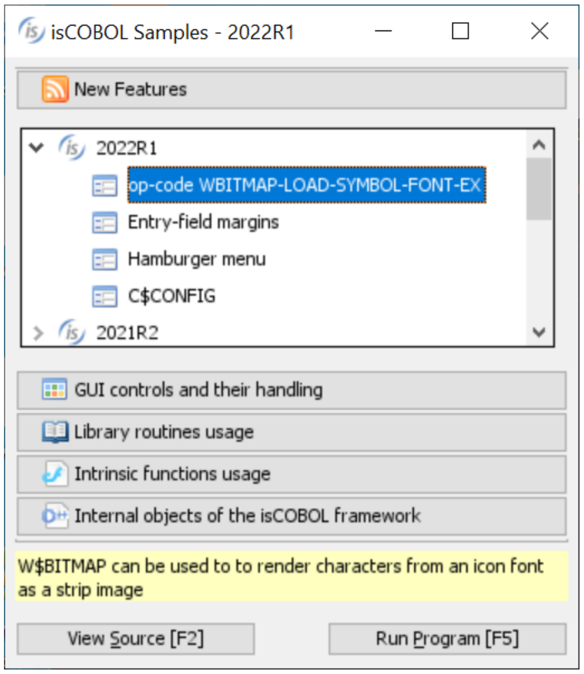
High DPI screens are common, and DPI scaling is essential to ensure the correct scaling of graphical screens across devices.

High DPI screens have higher pixel density compared to regular screens, and not scaling correctly could result in screen fonts being too small to be readable, or just not looking good. Applications are defined as DPI-aware if they can scale fonts and images to maintain the correct aspect-ratio and quality across devices.

When running an application in Windows, for example, if the application is not dpi-aware (known also as dpi-unaware), the operating system automatically manages the dpi scaling, but causes quality loss as a result of the upscaling. Starting from isCOBOL 2022 R2, the framework is dpi-aware and can scale to maintain a good quality output.

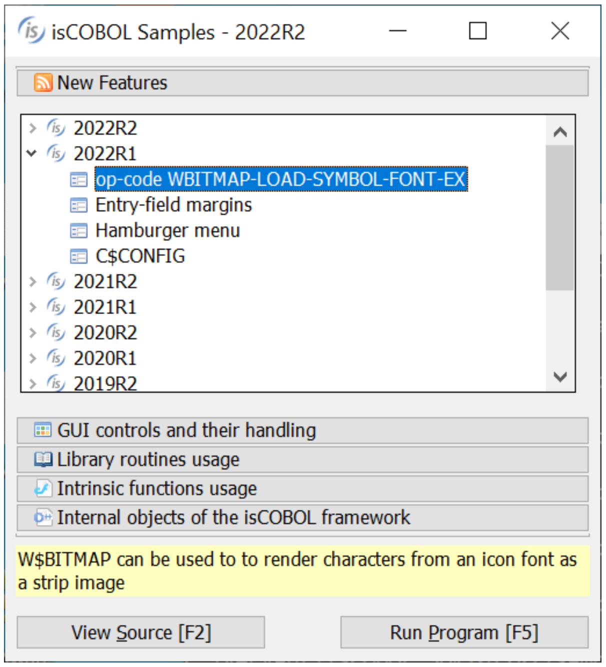
For example, running the Samples menu program on a 144 DPI display (a Windows system with 150% character size scaling) with isCOBOL 2022 R1 or previous versions, the screen looked like Figure 2, *Entry-field program not dpi-aware*.

Figure 2. Samples menu program not dpi-aware.



W$BITMAP can be used to to render characters from an icon font as a strip image

Figure 3. Samples menu program dpi-aware.



W$BITMAP can be used to to render characters from an icon font as a strip image

Running the same sample program on the same 168 DPI display with isCOBOL 2022 R2, the screen looks like Figure 3. *Entry-field program dpi-aware*.

Previous isCOBOL versions used a virtualized dpi-aware approach, increasing the pixel representation and losing quality, while the new isCOBOL version uses a real dpi-aware scaling, increasing the font sizes to keep a high-quality look. The same applies to images and icons.

If the previous virtualized dpi-aware approach is needed, it can be restored on Windows systems as follows: locate the isrun.exe (or the executable file used to start the program), right click on it and choose the “Properties” menu item, navigate to the “Compatibility” tab, click on “Change high DPI settings”, check the “Override high DPI scaling behavior” and choose “System” for “Scaling performed by:”.

New controls features

Several graphical controls that have been improved in the latest release:

- Check-Box and Radio-Button controls are now affected by a new configuration option named **iscobol.gui.icons_scaling**. When it is set to true, resizing by using the LM-ZOOM layout-manager automatically scales the icons of the screen's Check-Box and Radio-Button controls.
- Check-Box, Radio-Button and Push-Button controls now support the BITMAP-SCALE property, previously supported only on bitmap controls. When using bitmaps on the check-box, radio-button, and push-button controls with the new BITMAP-SCALE property set to 1, the bitmap is resized along with the window when using the LM-ZOOM layout-manager. This property affects all the bitmaps that are defined in Bitmap-Default, Bitmap-Disabled, Bitmap-Disabled-Selected, Bitmap-Number, Bitmap-Pressed, Bitmap-Rollover, Bitmap-Rollover-Selected and Bitmap-Selected for controls that have only a bitmap, or both bitmap and text with the position defined in the Title-Position property. A code snippet of controls with the new property is shown below.

```
03 check-box ...
    bitmap bitmap-handle bmp-handle bitmap-width 19
    bitmap-number 1 bitmap-rollover 2
    bitmap-scale  1.
 03 push-button ...
    bitmap bitmap-handle bmp-handle bitmap-width 19
    bitmap-number 3 bitmap-rollover 4
    bitmap-scale  1.
 03 radio-button ...
    bitmap bitmap-handle bmp-handle bitmap-width 19
    bitmap-number 5 bitmap-rollover 6
    bitmap-scale  1.
```

- Check-Box supports two new properties named `CHECK-ON-VALUE` and `CHECK-OFF-VALUE` allowing you to specify alphanumeric values associated with checked and unchecked states. This simplifies the management of check-boxes that need to store a different value than default numeric values 0 and 1, for example a flag that is stored with N and Y. Here's a code snippet of a check-box with a pic x variable set to N or Y:

```
77  w-flag  pic x.
...
03 check-box ...
   check-off-value "N" check-on-value "Y"
   value           w-flag.
```

  This new syntax is also useful when placing the check-box controls inside a grid using the statement "`Display Check-Box upon GridName (-1, 3)`" to show different values in the grid filter when the column includes a check-box.

- Radio-Button supports alphanumeric values in the property Group-Value, allowing a pic x variable to be used to hold the selection in a radio button group instead of the default numeric variable.

  Here's a code snippet of radio buttons with a pic x(n) variable than can be set to "A" "B" or "C":

```
77  w-type  pic x.
...
03 radio-button ...
   group 1 group-value "A" value w-type.
03 radio-button ...
   group 1 group-value "B" value w-type.
03 radio-button ...
   group 1 group-value "C" value w-type.
```

- Push-Button supports a new property named ROLLOVER-BORDER-COLOR to specify the border color of flat buttons when the mouse hovers on the control and can be used in conjunction with the existing properties Rollover-Background-Color and Rollover-Foreground-Color.

  A code snippet of a button with the new property is shown below:

```
03 push-button flat ...
   rollover-background-color  rgb x#A6F9DE
   rollover-foreground-color  rgb x#D51515
   rollover-border-color      rgb x#990066.
```

- Grid and List-Box controls support a new property named `EXPORT-FILE-OPEN` to automatically open an Excel file after it has been produced by the Export feature. This simplifies the opening of the files created on the client. Here's a code snippet of controls with the new property:

```
03  list-box ...
    export-file-name w-path
    export-file-format "xlsx"
    export-file-open 1.
03  grid ...
    export-file-name w-path
    export-file-format "xlsx"
    export-file-open 1.
```

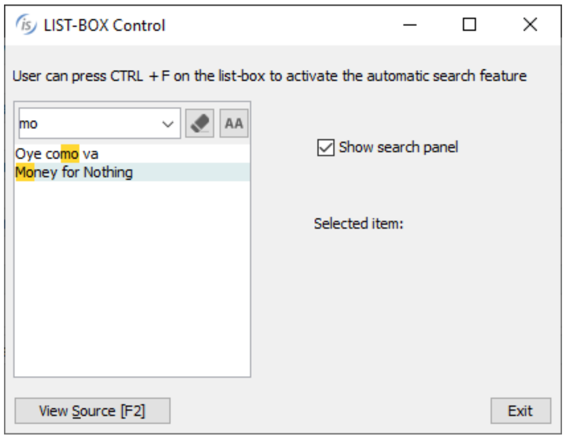
List-Box now supports the SEARCH-PANEL property using the same rules as defined in the grid control to manage the search feature. By default, this property is set to 0, making the search panel appear on top of the control only when the user presses the Ctrl-F keyboard shortcut. By setting the property to 1, the search panel will always be visible at the top of the List-box. To completely disable the search panel, the property can be set to -1. You can to change the shortcut key used to open the search panel with the keystroke configuration setting. To set the shortcut to Ctrl+G instead of Ctrl+F, for example, use the setting:

```
iscobol.key.*g=search=list-box
```

The following is a code snippet of a list-box with the new property set to 1 and Figure 4, *List-box search-panel in action*, showing the search panel used to filter the content of items loaded in the list-box.

```
 03  list-box ...
     search-panel 1.
```

Figure 4. List-box search-panel in action.



Tree-View supports a new MSG-MOUSE-CLICKED event, fired by both the standard Tree and Table-View tree when the NOTIFY-MOUSE style is set. This allows an action to be performed when clicking on an item instead of the standard double-click. When used on a Table-View, the special name EVENT-DATA-1 is set to the column number where the click happened.

Here's a code snippet to use the new event supported:

```
 03  tv1 tree-view table-view ...
     display-columns (1, 20)
     notify-mouse event tv1-event.
...
tv1-event.
    Evaluate event-type
    ...
    when msg-mouse-clicked
         if event-data-1 = 1 | click on first column
             ...
         end-if
    end-evaluate.
```

isCOBOL Runtime

Improvements to library routines and configuration settings have been added to the new isCOBOL runtime:

Improvements to library routines

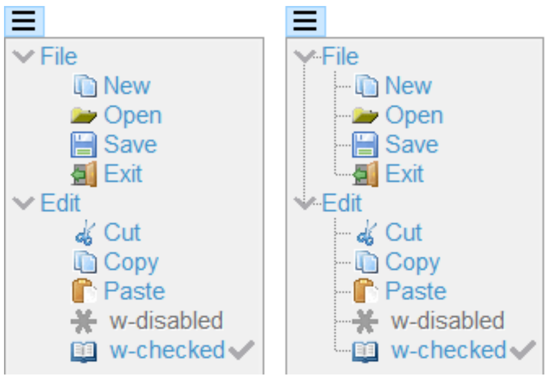
Several library routines have been improved in the 2022 R2 release:

- W$MENU library supports a new attribute named "show-lines" to show the lines connecting menu items in the hamburger menu. The following code snippet activates the new attribute:

```
 call "W$MENU" using wmenu-set-attribute "show-lines" "yes"
```

  The result when running the snippet is shown in in Figure 5, *Show-lines attribute of Hamburger menu*. The menu on the right has the new attribute set to “yes”.

  Figure 5. Show-Lines attribute of Hamburger menu.



- C$COPY and C$FSCOPY routines support an additional optional parameter to specify that the indexed file to be copied is encrypted. This works in conjunction with the JISAM file handler by using the same key set in the configuration option iscobol.file.encryption.key. The code snippet below shows the usage of the new parameter:

```
 call "C$COPY"   using src-path, dest-path, "I", 1
 call "C$FSCOPY" using src-path, dest-path, 1
```

- WIN$PRINTER library is now more flexible when creating PDF files using the –P PDF flag or saving a PDF file in the Print Preview. With previous releases it was mandatory to CALL the WINPRINT-SET-ATTRIBUTE op-code to set the attribute FONT_FOLDER or FONT_FOLDER_EMBED to embed fonts in the PDF file. In the current release, if these attributes are not set, isCOBOL will search for fonts in the default operating system's font folder: C:\Windows\Fonts on Windows, /usr/share/fonts on Linux and /Library/Fonts on MacOS, to embed fonts and use the Identity-H encoding in the generated PDF file.

New configurations

New configuration settings have been added in this release:

- **`iscobol.file.index.open_hook=ProgramName`** can be used to specify a hook program to be called before a file is opened, and allows the file path or file handler class to be modified on the fly. The ProgramName must declare two parameters received in linkage section as follows:

```
linkage section.
77  file-path   pic x(300).
77  file-class  pic x(300).
procedure division using file-path, file-class.
```

  The hook program is executed just before the file is opened and, if necessary, can update the value of the two parameters to change the file path or the file handler. This flexible solution can be used, for example, to easily migrate programs from a Fat-Client to a Thin-Client architecture, where the "local working files" are not on the client PC, but are located on the server, and files might have to be kept separate for different users.

- **`iscobol.floating_point_format=ibm_hfp`** to store data-items with usage 'float' or 'double' using the IBM hexadecimal floating-point format (known also as HFP by IBM), instead of the default "ieee_754", the technical standard for floating-point arithmetic established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). In comparison to IEEE 754 floating point, the HFP by IBM format has a longer coefficient, and a shorter exponent.

- `iscobol.gui.nested_embedded_proc_check=true` to check for nested accepts in embedded procedures. When this configuration option is set, a specific exception is thrown. This helps the developer easily identify and correct any nested accepts found in the code.
- `iscobol.file.index.ctshmemdir=directory` to set the sharedmem directory for the CTREEJ file handler. On Unix platforms the c-treeRTG server shares a directory with the c-treeRTG clients when the communication is performed using the shared memory protocol. When using multiple c-treeRTG servers, it's important to configure the SHMEM_DIRECTORY in each server's ctsrvr.cfg configuration files, and set the same value in this new isCOBOL configuration setting, so that every server uses a separate folder for SHAREMEM communication.
- `iscobol.file.index.endiancheck=false` to skip the byte endianness check made by CTREEJ file handler during connection. By default, the check is made to ensure that the byte endianness of both client and server is the same. In cases where this check is not necessary, such as when accessing files not containing native numeric data types, the check can be relaxed so the connection can succeed.

Additionally, it's now possible to configure the exception values returned when the user just presses a letter without any additional special key. This helps in situations such as when accepting a screen containing only controls not used for editing, for example push-buttons, allowing the user to quickly choose an option by just pressing a letter without the need to press it in combination with Alt or Ctrl. For example, configuring:

```
iscobol.key.a=exception=101
iscobol.key.b=exception=102
```

when the user presses the "a" key, the program intercepts the exception-value 101, and 102 is received when pressing the "b" key.

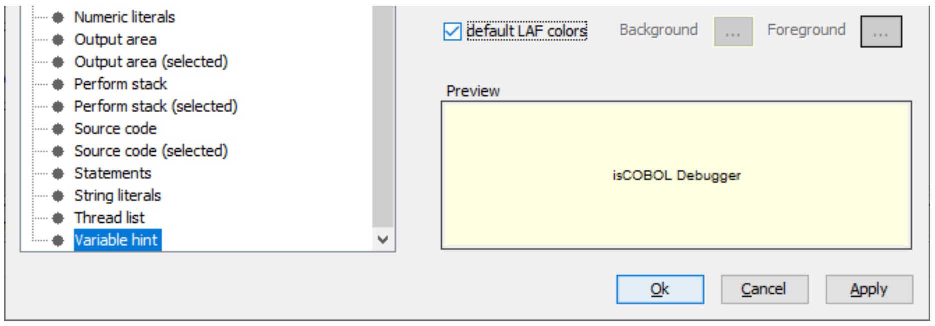
isCOBOL Debugger

The isCOBOL Debugger has been enhanced in the 2022 R2 release to improve the usability when using different LAFs. These are the features added in the new release:

- colors configured in Debugger can now follow the standard LAF color settings

The debugger's "Fonts and Colors" window, from the menu's Settings/Customize option, is where you would set the different fonts and colors for different parts of the Debugger. In 2022 Release 2, you can set the "default LAF colors", and the result will be different when running on different operating systems or when running with a different LAF. The colors will then be more coherent, for example the Hint displayed when the mouse is on a data item, will look the same as the Hint displayed for the buttons. The new setting is shown in Figure 6, *Debugger LAF color*.

Figure 6. Debugger LAF color.


- Numeric literals
- Output area
- Output area (selected)
- Perform stack
- Perform stack (selected)
- Source code
- Source code (selected)
- Statements
- String literals
- Thread list
- Variable hint

☑ default LAF colors Background Foreground

Preview

isCOBOL Debugger

- new option -a for Debugger's infostack command to show all available info

The **infostack** is a command that can be used in the command-line to list all paragraphs and programs involved in the stack. With the new option ***–a***, the source file and line number are shown in the output. For example this is the output of the command with the new –a option executed while debugging:

```
+ MAIN [THREAD_BRIDGE] source/THREAD-BRIDGE.cbl:34
  + MAIN [ISCUSTOMER] source/ISCUSTOMER.cbl:265
    + AFTER-ACCEPT [ISCUSTOMER] source/ISCUSTOMER.cbl:291
      + READ-NEXT [ISCUSTOMER] source/ISCUSTOMER.cbl:424
```

WebClient

isCOBOL WebClient 2022 R2 is Veryant’s solution for running desktop applications in a browser, and it has been updated to support Java 17, to complete the process started with the 2022 R1 release to certify all Veryant products with all current LTS Java versions: 1.8, 11 and 17.

The new version of WebClient has been optimized to better handle low speed and high latency connections. It now also supports window transparency, and has improved touch screen integration. The session recording and mirroring and new Admin console are among the many other improvements.

isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. As of version isCOBOL 2022 R2, the HTTPClient class can consume HEAD requests to retrieve only header fields.

HTTPClient

HTTPClient is a class that enables COBOL programs to interact with Web Services. This class has been updated with these new method signatures:

```
public void doHead (String strUrl, HTTPData.Params p)
```

```
public void doHead (String strUrl)
```

The new methods will perform an HTTP HEAD request on the provided URL and using optional HTTP parameters.

The HTTP HEAD method requests HTTP headers from the server as if the document was requested using the HTTP GET method. The only difference between HTTP HEAD and GET requests is that for HTTP HEAD, the server only returns headers without the body.

The HTTP HEAD method is much faster than the HTTP GET method because much less data is transferred in HEAD requests. Browsers use the HEAD method to update information about cached resources to check if the resource has been modified since the last time it was accessed. If the resource has not been modified, browsers reuse the local copy without issuing a new request. Otherwise, they request an updated version of the resource with a GET request.

The HEAD method can be also used to check if a file on the server exists, without actually downloading it if not really necessary.

Additional improvements

isCOBOL 2022 R2 release improves the JUTIL utility.

A new option -make is now supported to create an empty Jisam file from the xml dictionary. This approach is equivalent to the ctutil -make option already supported by ctutil for c-TreeRTG.

Here's an example of the new option used to create an empty Jisam file named "products" based on its definition stored in "product.xml":

```
jutil -make products products.xml
```

Note that "products.xml" is the file dictionary that describes the indexed file with all its details and is created by isCOBOL's compiler when using the -efd option.