isCOBOL™ Evolve

isCOBOL Evolve 2023 Release 2 Overview

isCOBOL Evolve 2023 Release 2 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2023 R2.

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

isCOBOL Evolve 2023 R2 GUI controls now support drag-and-drop functionality, and the grid control has been enhanced with load-on-demand and new filtering options.

IDE projects can now be run using a specific Java version.

The new TurboRun feature can be used to significantly speed up sequential batch programs execution.

Details on these enhancements and updates are included below.

GUI enhancements

Several improvements to graphical user interface components have been implemented in this release, providing the ability to use drag-and-drop actions on GUI controls, enhancing the grid control with new properties and events and additional capabilities in images integration.

Drag-and-Drop support

In computer graphical user interfaces, drag and drop is a pointing device gesture in which the user selects a virtual object by "grabbing" it and dragging it to a different location or onto another virtual object. The basic sequence involved in drag and drop is:

- Move the pointer to the object.
- Press and hold down the button on the mouse or other pointing device to "grab" the object.
- "Drag" the object to the desired location by moving the pointer to that location.
- "Drop" the object by releasing the button.

isCOBOL compiler now supports a new property on most controls named `drag-mode` that allows you to specify which kind of action is enabled. If the property is not set, the default value 0 is assumed, and no actions are enabled. To enable actions the following values can be used:

1 = dragging action is enabled

2 = dropping action is enabled

3 = both dragging and dropping actions are enabled

When the drag action is enabled, the control receives a new event named `MSG-DRAG` and when the drop action is enabled, the control receives a new event named `MSG-DROP`.

Currently the controls that fully support this property are: BITMAP, CHECK-BOX, COMBO-BOX, DATE-ENTRY, ENTRY-FIELD, GRID, LABEL, LIST-BOX, PUSH-BUTTON, RADIO-BUTTON, TREE-VIEW.

For example, a screen section defined as:

```
03 gd-article grid ...
   drag-mode 3
   event GD-ARTICLE-EVENT.
03 gd-order grid ...
   drag-mode 3
   event GD-ORDER-EVENT.
03 ef-article entry-field ...
   drag-mode 1
   event EF-ARTICLE-EVENT.
03 pb-delete push-button ...
   drag-mode 2
   event PD-DELETE-EVENT.
```

contains 2 grids with drag and drop enabled, an entry-field with just dragging enabled and a push-button with just dropping enabled. In this example, a row from a grid can be moved to another grid by dragging it from one grid to the other, and the number in the Quantity column can be increased when dropping an existing row into the target grid.

The event procedure shown below handles the logic of these actions by managing the two new events:

```
GD-ARTICLE-EVENT.
    evaluate event-type
    when MSG-DRAG
         inquire gd-article(event-data-2 1) cell-data gdr-article
         ...
    when MSG-DROP
         ...
         modify gd-article(idx 2) cell-data gdr-qty
    end-evaluate.
EF-ARTICLE-EVENT.
    if event-type = MSG-DRAG
       inquire ef-article value ef-dragged-element
       modify ef-article cursor -1
       set drag-from-ef-article to true
    end-if.
PD-DELETE-EVENT.
    if event-type = MSG-DROP
       if drag-from-gd-order and event-data-2 > 1
          perform DROP-TO-PB-DEL
       end-if
    end-if.
```

As shown in Figure 1, *Drag allowed*, the mouse shows the plus icon to inform the user that it's possible to drop the content being dragged. In Figure 2, *Drag prohibited*, the mouse shows the deny icon to imply that it's not possible to drop the dragged content on the control. Both screenshots were taken from running the installed sample for this new feature that contains the code shown in the previous snippets.

Figure 1. Drag allowed.

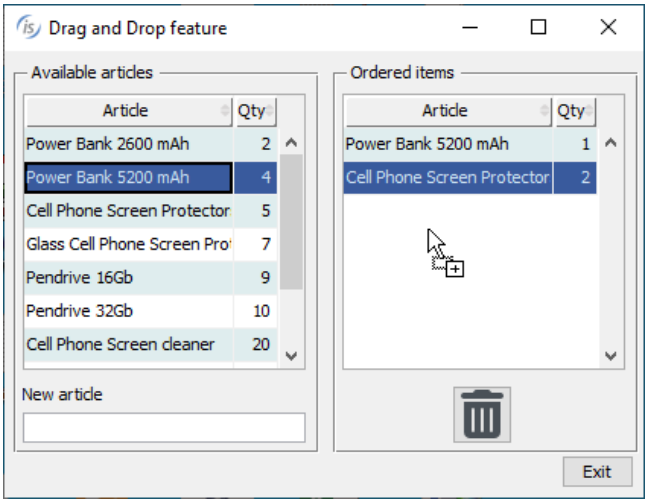
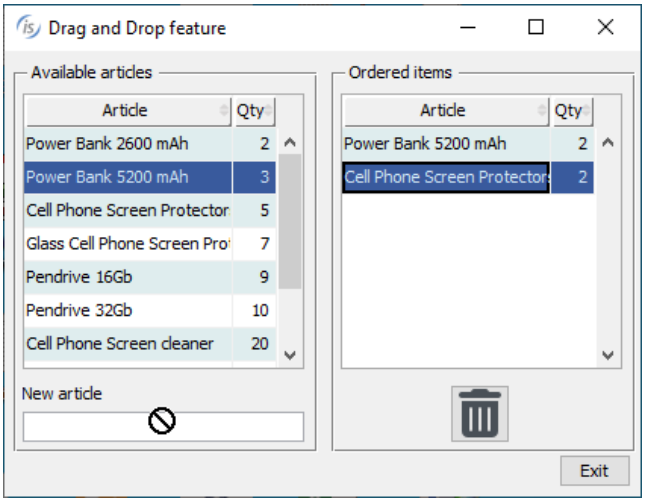


Figure 2. Drag prohibited.



Grid enhancements

The grid control has been enhanced in 2023 R2 to support the load-on-demand feature used to dynamically load content as the user scrolls down in the grid. This approach is beneficial when a grid contains a large number of rows by loading a smaller amount at a time thus reducing the time and resources required to fetch the data.

A grid loaded with the load-on-demand feature is initially filled with a given number of records for fast initial loading, and as the user scrolls down in the grid the new event MSG-LOAD-ON-DEMAND is fired and the application can respond by loading additional rows seamlessly, reducing the application load and delivering a smoother experience for the user.

A specific example is installed in the sample\issamples\s-gui folder, and the following code is a snippet taken from GRID-ON-DEMAND.cbl source:

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

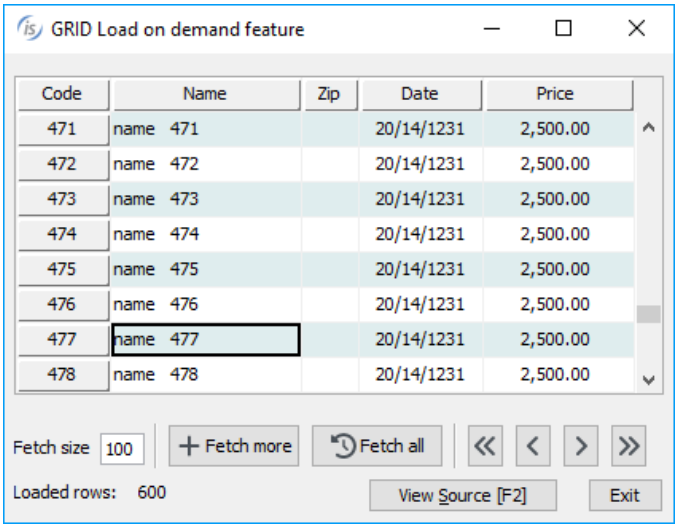
This grid definition shows the new property lod-threshold (Load-On-Demand threshold) used to specify the percentage of scrolling in the grid that triggers the MSG-LOAD-ON-DEMAND event. When the user scrolls past 80% of the initially loaded records, the event is fired.

The code snippet below shows the event procedure paragraph that manages the response logic for the new event:

```
GD-EVT.
    evaluate event-type
    when MSG-LOAD-ON-DEMAND
         perform fetch-size times
            ...
            modify gd record-to-add gd-rec
         end-perform
    end-evaluate.
```

In Figure 3, *Load on demand feature*, the program in execution shows the grid loaded during scroll-down.

Figure 3. Load on demand feature.



A new filter type has been added to enable editing the value for filters in selected columns. Using an entry field instead of a check list-box for a filter can be beneficial when the content of a column contains many different values, making a list too long to prove useful. The new filter is declared using the value 3 in the existing property FILTER-TYPES. It can be also mixed with previous types, making it possible to have specific filter types for each column. The panel with the filters' entry-fields is shown between the heading line and the first grid line when the user clicks on the funnel icon.

There are also two new properties that can be used with the inquire or modify statements:

- `filter-panel` to hide the panel with fields if set to 0 (default 1)
- `column-filter` to save and restore the column filter value

The following snippet shows how the properties can be used:

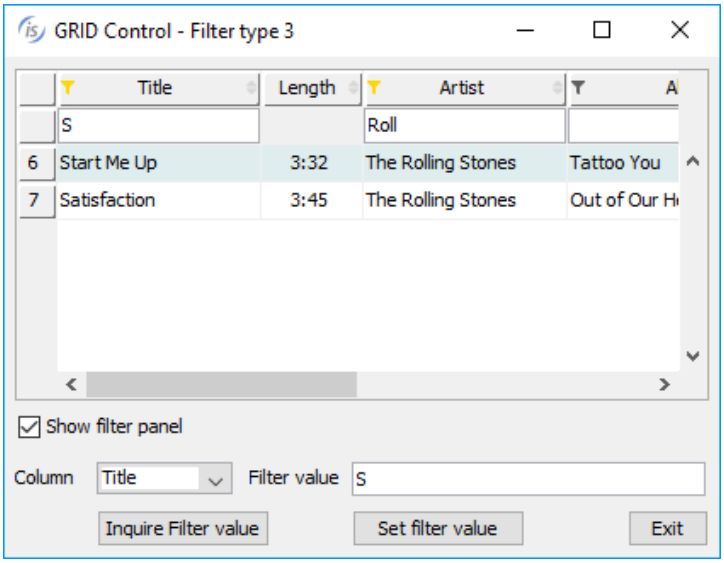
```
modify gd(1, num-col) column-filter wrk-filter
modify gd filter-panel 0
```

The following code snippet:

```
03 gd grid ...
   filterable-columns
   filter-types (0, 3, 0, 3, 3, 2, 3, 1).
```

declares a grid with the new filter type used on columns 2, 4, 5 and 7. The result of the program in execution is shown in Figure 4, *New filter type in grid*.

Figure 4. New filter type in grid



A new property named CELL-SECURE can now be used in MODIFY and INQUIRE statements to manage the data in the cell as an entry-field with SECURE style. This is necessary in password fields that need to have the "hide/show" behavior dynamically. When the property is set to 1, the cell identified by Y and X coordinates assume the secure feature, making the column shown as asterisks "*****".

This is a code snippet to set the cell-secure in the cell of row 3 and column 2 and inquire the same property in the cell of row 5 and column 2:

```
modify  gd(2, 2) cell-secure 1
inquire gd(5, 2) cell-secure w-secure
```

Image improvements

The bitmap control now supports two new properties that can be used with the INQUIRE statement to retrieve the raw image dimensions:

- `bitmap-raw-height` returns the height of the image in pixels
- `bitmap-raw-width` returns the width of the image in pixels

These values are also returned by W$IMAGESIZE routine, but instead of passing the handle of an image loaded with W$BITMAP they can be inquired directly from the bitmap control.

This is a code snippet that shows how to use the new properties:

```
inquire bmp-prod bitmap-raw-height wraw-height
inquire bmp-prod bitmap-raw-width  wraw-width
```

These properties are useful when images are loaded dynamically into fixed-size controls. The program can check to see if the loaded image is larger than the control's size, and apply scaling accordingly to ensure that the whole image is visible.

The W$SCALE routine has been improved by supporting a new value named WSCALE-RESIZE-NONE in the scaleMode parameter, allowing the program to create a new image with different dimensions from the original one, but without scaling it. In conjunction with the scaleAlign parameter, it can be used to align the resulting image as needed.

This is a code snippet that uses the new parameter value in the routine:

```
      move 50                     to wrk-cells-newWidth
      move 18                     to wrk-cells-newHeight
      move WSCALE-RESIZE-NONE     to wrk-scale-mode
      move WSCALE-AL-MIDDLE-CENTER to wrk-scale-mode
      call "W$SCALE" using h-bitmap
                           wrk-cells-newWidth
                           wrk-cells-newHeight
                           hWin
                           wrk-scale-mode
                           wrk-scale-align
                     giving h-bitmap-scaled
```

isCOBOL IDE

The isCOBOL IDE 2023 R2 includes a new feature that allows you to run or debug programs using specific JRE versions for each project.

Configuring JRE in IDE projects

By default, isCOBOL IDE runs with the Java JDK chosen during installation, with version 11 being the minimum allowed version. When compiling, the --source and –target options are set to 1.8 by default, making the compiled .class runnable starting from JRE 1.8.

Applications may need to be run using different Java releases for any number of reasons, for example when using Object Oriented Programming with dependencies on classes compiled with specific JDK releases. Different Java distributions (for example Oracle JDK or OpenJDK) can be selected as well.

To fully support the execution of Cobol applications with different Java releases, isCOBOL IDE 2023R2 reads the list of installed JREs in Eclipse's preference page, as shown in the Figure 5, *Preferences of Installed JREs*.

In the isCOBOL Settings / Java Runtime properties page, it's now possible to specify one of the installed JREs to be used when running the project's programs, as shown in Figure 6, *Java Runtime set on project properties*.

Each project can use its own Java version or distribution.

Figure 5. Preferences of Installed JREs

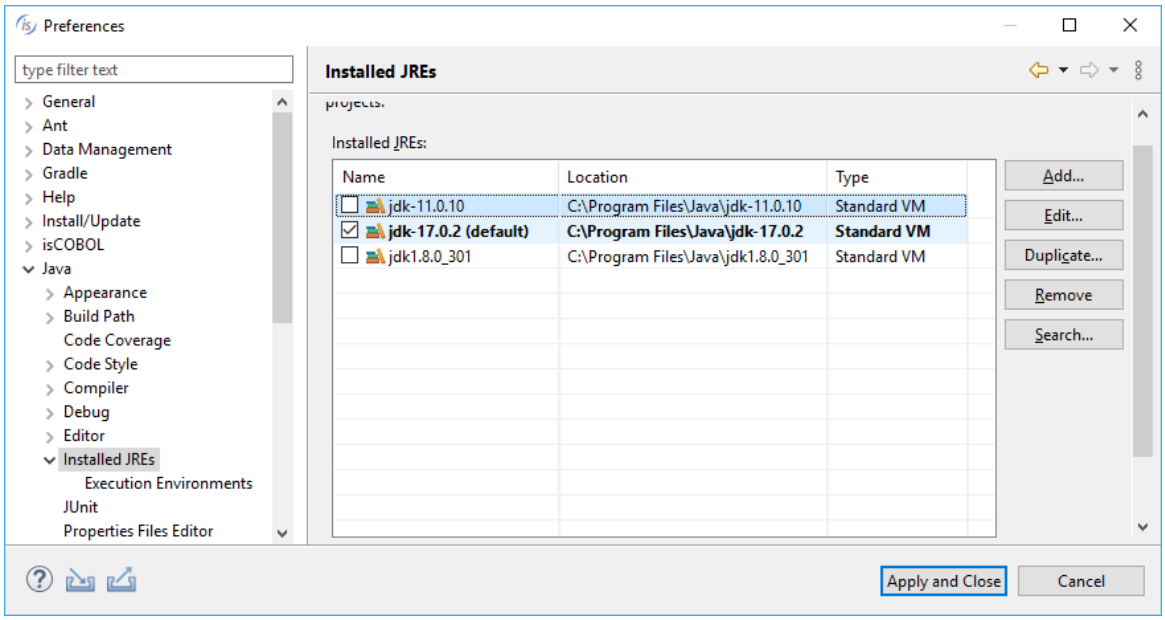
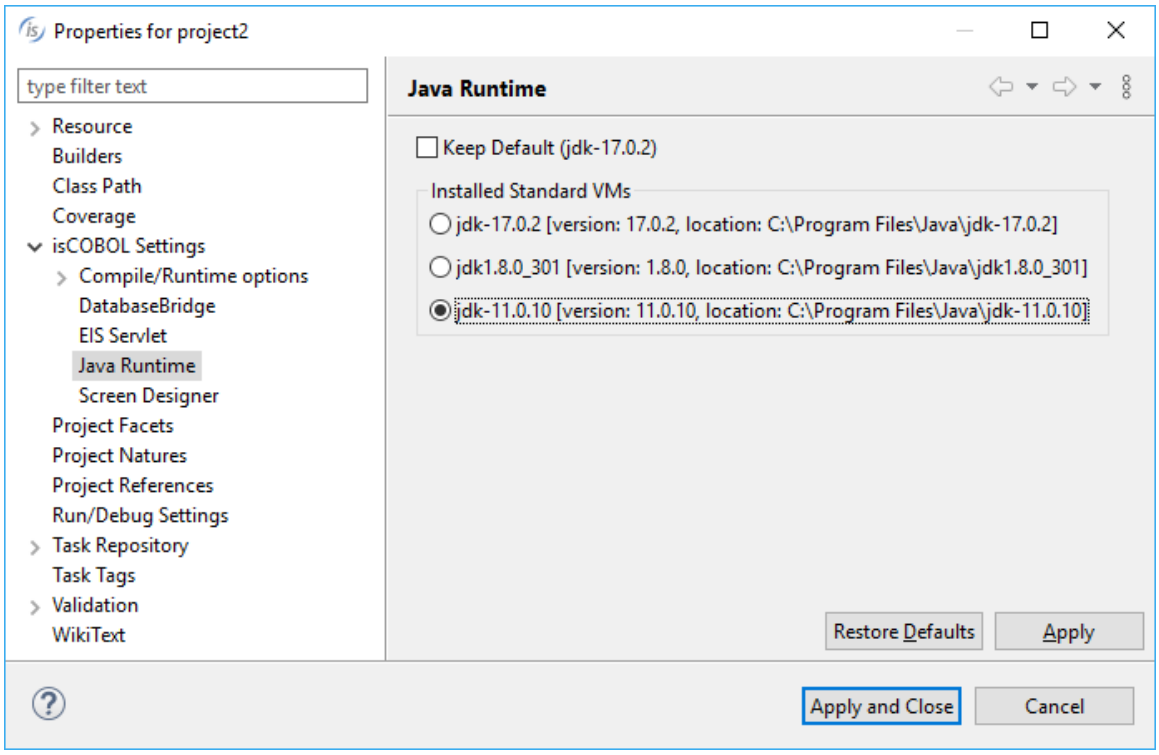
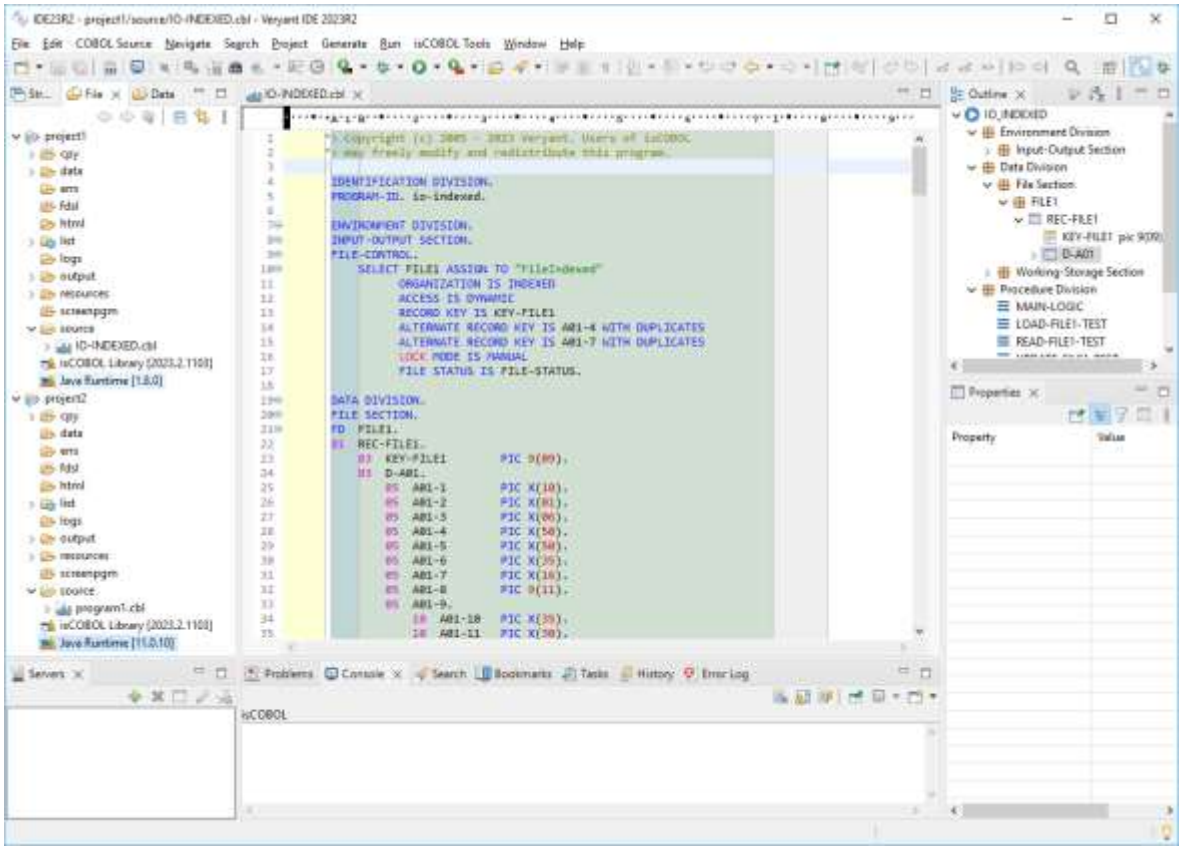


Figure 6. Java Runtime set on project properties



The chosen version of the Java runtime is shown in the workspace as the last item of each project tree after the isCOBOL library release. For example, in Figure 7, *Java Runtime version* shows project1 that is configured to run programs with Java 1.8.0, while project2 is configured to run programs with Java 11.0.10.

Figure 7. Java Runtime version



isCOBOL Compiler

The isCOBOL 2023 R2 compiler includes new compiler options to simplify the migration from other COBOL dialects and other improvements.

Improved Compatibility with other COBOLs

In this release, two new compiler options have been added to increase compatibility with other COBOL dialects.

The new -ccmf option enables MicroFocus compatibility in numeric literal parameters passed in a CALL statement. For example, in the following code snippet:

```
CALL "PROGB" USING BY VALUE 123
```

the numeric parameter 123 is passed as if it is in a usage display variable data item PIC 9(3) that uses 3 bytes if the option is not used. Instead, using the new compiler option, the parameter is passed as a signed numeric item USAGE BINARY, as if it is a PIC S9(9) BINARY that uses 4 bytes.

This option is useful when the called program does not implement dynamic management of parameters, for example calling the C$DARG and C$PARAMSIZE library routines to determine what kind of parameters have been passed and how many bytes are received from the calling program.

The new -cmc option allows you to assume WITH CONVERT in every applicable MOVE. This option helps in migrations from Cobol-IT where a move like:

```
MOVE VARX TO VARNUM
```

where variables are defined as different pictures/size, such as:

```
01  VARX   PIC X(6).
01  VARNUM PIC 9(3).
```

and the VARX data item contains “123.00”, with the option enabled, the move statement will set the VARNUM data item to 123. In other words, the MOVE from alphanumeric to numeric data item is treated as if it was written with the WITH CONVERT clause:

```
MOVE VARX TO VARNUM WITH CONVERT
```

Other improvements

The current list of compiler options set by default when compiling, to ensure better performance and compatibility, are: -b –cghv -g -oe

If compiling with the –cp option to enable full support of C pointers, the -m1 option is added to the default options as well.

The compiler now supports the "-no-" prefix before the option name on the command line to remove an option set by default. For example, if a program needs to be compiled without the default –b option, use the command line:

```
iscc -no-b -d PROGNAME.cbl
```

To check which compiler options have been used at compile time on an existing .class, you can run the command:

```
iscrun -info PROGNAME
```

which will print the following output:

```
C:/myapp/classes/PROGNAME.class: compiled with isCOBOL build #1105, minimum
required #1103, debug
Compile flags: -g -oe -cghv -d
Compiled on: June 16, 2023 11:30:23 AM
```

Starting from 2023R2, the compilation date and time is added to the above list of information.

The same information shown above can also be retrieved at runtime using the new library function `compiled-info`, and the following code:

```
MOVE $COMPILED-INFO TO VARX
```

or:

```
MOVE FUNCTION COMPILED-INFO TO VARX
```

will set the VARX variable to the same data shown by running the "iscrun –info" command.

The new `-classinfo` option has been added to the `display` Debugger command to display the same metadata, i.e.:

```
display -classinfo
```

A new compiler configuration option named `iscobol.compiler.debug.replaced_source` has been added to determine what code will be shown in the Debugger when debugging a program.

When compiling with the configuration set to "true" (default), the replaced source will be shown by the Debugger.

When compiling with the configuration set to "false", the original source will be shown by the Debugger.

This option can be useful when the compiled source is modified by an external tool before being compiled, for example when using the integrated PreProcessor by setting the configuration iscobol.compiler.custompreproc or iscobol.compiler.regexp that replaces the source code using regular expression rules.

For example, compiling the sample installed in the "sample\compiler-pre-process\custom-log" folder without the new compiler configuration, the default value true is assumed, and the Debugger will show the code in Figure 8, *Debugger with replaced code*. If compiling using `iscobol.compiler.debug.replaced_source=false`

the Debugger will show the code in in Figure 9, *Debugger with original code.*

Figure 8. Debugger with replaced code.

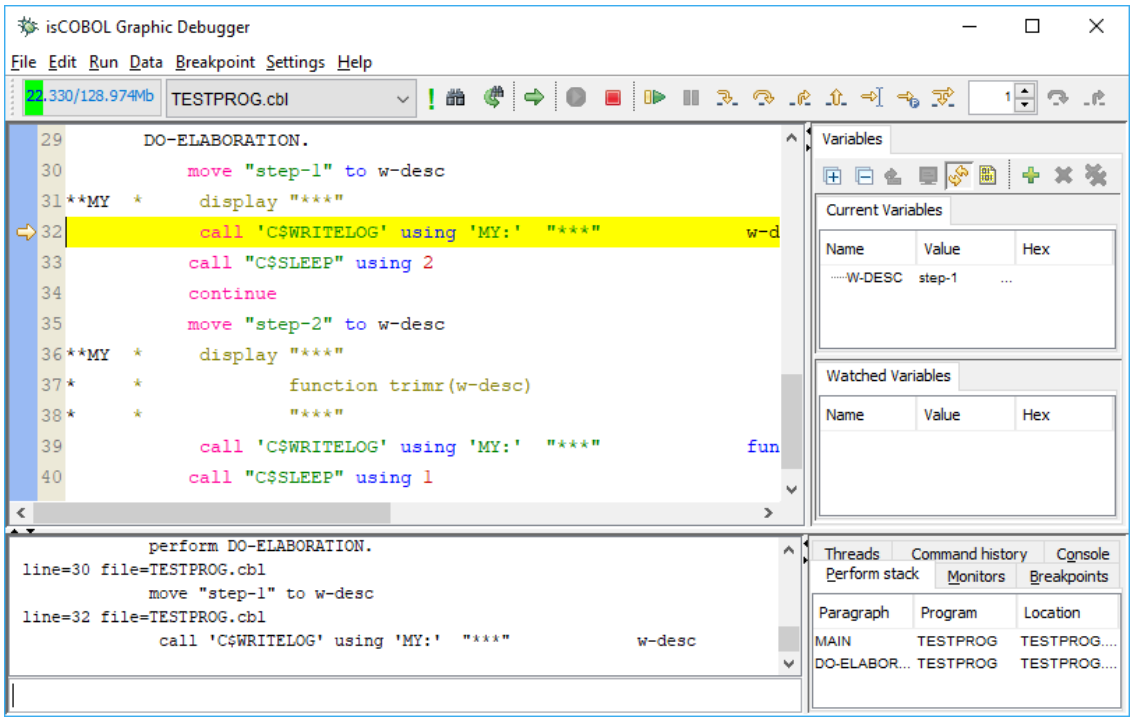
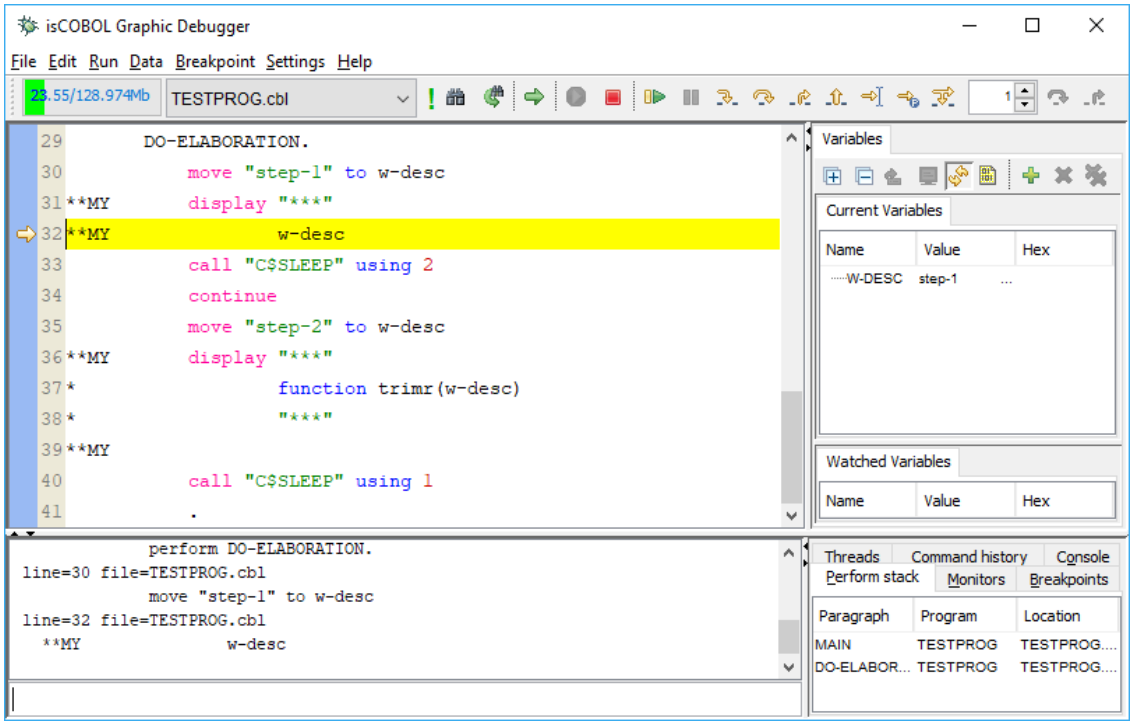


Figure 9. Debugger with original code.



IsCOBOL Runtime

The isCOBOL 2023 R2 runtime includes new configuration settings to customize the behavior for JDBC data access and environment.

New configuration settings

New supported configurations settings have been added in this release:

- `iscobol.env.trunc_on_null`=true can be used to truncate an accepted environment variable to the first low-value when storing it in the COBOL variable. This can be used as a compatibility option with MicroFocus and ACUCOBOL-GT. If set to the default value of false the full value, including low-values, will be returned.
- `iscobol.esql.picx_handling`=n to set the mapping of database character fields to PIC X host variables. If set to the value of 2, the runtime detects the database field type, and sets the COBOL data item accordingly. When set to 1, the runtime assumes all database fields to be binary. This is an uncommon situation, and should be used rarely. When set to the default value of 0, this feature is disabled, and the data items will be handled according to the $SQL compiler directive. With previous isCOBOL versions, or when this setting is 0, it is the programmer's responsibility to describe the data items in the COBOL working-storage using the $SQL compiler directive to allow the runtime to correctly map binary database fields, otherwise the runtime defaults to character fields.
- `iscobol.national.acu_compatibility`=true to use standard Ascii characters in 2 bytes in PIC G or PIC N when compiled using the -cndbcs option. The default value is false to continue using standard Ascii characters in 1 byte, and it is useful for MicroFocus compatibility. For ACUCOBOL-GT compatibility, a value of true should be used.

A new method has been added in com.iscobol.java.IsCobol class to check if a program is available. This is useful in environments where Java applications need to check the existence of a Cobol program before calling it. The signature of the new method is:

```
public static boolean exists(String programName)
```

IsCOBOL Application Server

The isCOBOL Application Server 2023 R2 includes a new feature named TurboRun for faster execution of COBOL batch programs, and A$ routines have been enhanced.

TurboRun feature

The new TurboRun feature improves performance of COBOL batch programs that are executed sequentially by running them in the Application Server Java Virtual Machine, saving the cost of initializing a JVM for each application. Using the Application Server JVM boots performance even further, since it uses the cache and the JIT (Just-In-Time) optimizations that are already running.

TurboRun is a native lightweight task that starts the program by asking the Application Server to run the program.

To use this feature, start the Application Server with the new -tr option, for example:

```
iscserver -tr
```

that produces the following output:

```
Application Server (TurboRun) started and listening on port 10995
```

Then you can run a command like:

```
trun -c /path/to/configuration_file PROGRAM_NAME
```

that will start much faster than a command like:

```
iscrun -c /path/to/configuration_file PROGRAM_NAME
```

As shown in the output, the port number 10995 is the default value used for this new service. It can be changed using the –trport option, for example to use port number 11300, you can use the following command:

```
iscserver -tr -trport 11300
```

and then trun processes can be started with the –port option, similar to the iscclient:

```
trun -port 11300 -c /path/to/configuration_file PROGRAM_NAME
```

The new –tr option can be used in conjunction with the previous -as, -fs or –hs options since all they use different port numbers for the communication.

A$ routines

Library routines for the management of connected thin clients have been improved as follows:

- A$USERINFO now supports the optional TID (Thread ID) parameter in the AUSERINFO-CLEAR op-code. This is consistent with other op-codes (AUSERINFO-GET and AUSERINFO-SET) that allow you to retrieve or set the customized info for another TID. The following code snippet shows the usage:

```
call "A$USERINFO" using auserinfo-clear,
                        th-id
```

- A$LIST_USERS, A$CURRENT_USER and A$GET_USER can now optionally return the custom information bound to each client, if present. Now the additional call to the A$USERINFO to retrieve the custom information bound to the thread ID is no longer needed. This is the code snippet of the new usages:

```
call "A$LIST-USERS" using listusr-next, usrlist, usr-id,
                          usr-name, usr-addr, usr-pcname,
                          usr-tid, usr-prog, usr-type,
                          usr-logon-time, usr-cust-info
...
call "A$CURRENT-USER" using usr-id, usr-name, usr-addr,
                            usr-pcname, th-id, usr-prog,
                            usr-type, usr-logon-time
                            usr-cust-info
...
call "A$GET-USER" using usr-tid, usr-id, usr-name,
                        usr-addr, usr-pcname, usr-prog,
                        usr-type, usr-logon-time,
                        usr-cust-info
```

- A$USER_INFO and A$GET_USER now return a specific error number -2 if the passed thread ID is invalid, and can be used in a COBOL application to check if a TID is still valid. This is the code snippet of the new usages:

```
call "A$USERINFO" using auserinfo-get, info-retrieved, th-id
                  giving return-code
...
call "A$GET-USER" using usr-tid, usr-id, usr-name,
                        usr-addr, usr-pcname, usr-prog
                  giving return-code
if return-code = -2
  ...
```

Additional improvements

isCOBOL 2023 R2 release contains additional improvements in the DatabaseBridge, EIS products and in the Utilities.

Database bridge

EDBI routines for Microsoft SQL Server now map the COBOL file's RECORD KEY to a PRIMARY KEY instead of a UNIQUE INDEX. Using a PRIMARY KEY instead of a UNIQUE INDEX provides a more standard SQL code.

In addition, a new compiler property `iscobol.compiler.easydb.not_null_columns`=true (or using the new `-nn` option on the `edbiis` executable command) insures the generated EDBI routines will add the NOT NULL clause to all fields when creating the table. This will force external SQL clients to set all the fields when they insert or update records.

HTTPHandler class

A new method in HTTPHandler class has been added to pass a custom mime type to the displayText method:

```
void displayText( text, mimetype )
```

Now you can pass "text/xml" or "text/json" instead of the default "text/plain" MIME type. This is useful if the COBOL program creates the json or xml string without using a structure declared with the IS IDENTIFIED clause.

Utilities

New configurations have been added to customize the font used by different utilities:

- `iscobol.as.panel.custom_font=<name>-<size>` to set the font used by isCOBOL Server Panel
- `iscobol.jdbc2fd.custom_font=<name>-<size>` to set the font used by JDBC2FD utility
- `iscobol.isl.custom_font=<name>-<size>` to set the font used by ISL utility
- `iscobol.ismigrate.custom_font=<name>-<size>` to set the font used by ISMIGRATE utility

In addition, the icons used in isCOBOL utilities have been updated to use Font Awesome icons, providing a more modern look and better integration when running them in the web via WebClient. For example, Figure 10, *New icons in isCOBOL Panel*, shows the new look used by isCOBOL Panel.

Figure 10. New icons in isCOBOL Panel.