isCOBOL™ Evolve

isCOBOL Evolve 2019 Release 1 Overview

isCOBOL Evolve 2019 Release 1 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2019 R1.

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

isCOBOL 2019 R1 is now certified to run with both Oracle Java 11 and OpenJDK 11 and introduces new features to simplify distribution and updates of isCOBOL Thin Client applications.

The 2019R1 Release introduces a new responsive layout-manager, and GUI controls have been upgraded with new features and compatibility options.

The isCOBOL IDE is now based on the latest Eclipse 2018-09 IDE, supports JDK 11, and can now import and support COBOL-WOW projects.

The isCOBOL EIS Web Service bridge has been improved and now allows you to embed custom user code in the automatically generated bridge code.

Details on these enhancements and updates are included below.

Support for Java 11 and OpenJDK

isCOBOL Evolve 2019 R1 now supports Oracle Java 11 and OpenJDK 11, the latest releases of Java currently available, bringing isCOBOL up to date with the Java ecosystem.

isCOBOL and isCOBOL IDE installation setups now give you the option of selecting an OpenJDK installation, and all Veryant products are certified to run using Oracle JDK 11 and OpenJDK 11 for maximum flexibility.

The isCOBOL updater tool, isUPDATER, has been expanded in response to Oracle's decision to drop support for Java Web Start, and now simplifies the startup of isCOBOL Thin Client applications.

Java Web Start (JAWS) is a framework developed by Sun Microsystems (now Oracle) that allows users to start application software for the Java Platform directly from the Internet using a web browser. Some key benefits of this technology include seamless version updating for globally distributed applications and greater control of memory allocation to the Java Virtual Machine. As of JDK9, Java applets are deprecated by Oracle with Java Web Start being the intended replacement. In March 2018, Oracle announced it will not include Java Web Start in Java SE 11 (18.9 LTS) and later. Developers will need to transition to other deployment technologies.

Since isCOBOL now supports Java 11 there is a need to replace the Java Web Start's functionality for COBOL application startup with the isCOBOL automatic updater, isUPDATER.

The newest isUPDATER supports the HTTPS protocol, which can be configured using the following configuration properties:

- `swupdater.net.ssl.trust_store=keystore` to set the keystore
- `swupdater.net.ssl.trust_store_password=password` to set the password of the keystore
- `swupdater.http.ignore_certificates=true` (default is false) to ignore invalid certificates

Media Type application

Veryant is in the process of registering two new mime types as applications of vnd.veryant.thin at www.iana.org. These will enable the automatic execution of isCOBOL Thin Client and the isCOBOL updater tool.

These file suffix registrations, done from the isCOBOL installation setup screen, will simplify the execution and the automatic update for the end user. The registered files contain all the isCOBOL properties needed to set up the desired execution of isCOBOL THIN and isUPDATER technologies on the client side.

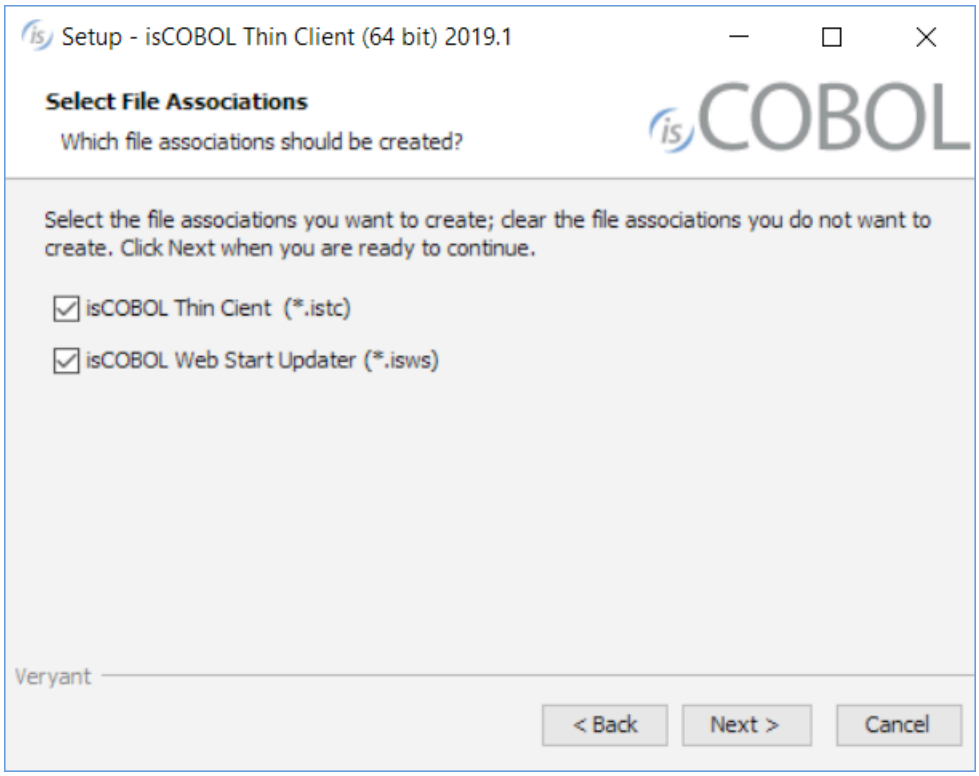
When a registered file is downloaded, a user can execute it immediately without additional configuration steps.

The new file extension associations are:

- **.istc** to run the isclient utility (isclient -lc %1)
- **.isws** to run the isUPDATER (isupdater -c %1)

The new isCOBOL 2019R1 installation screen shown in Figure 1, *isCOBOL file associations,* prompts you to define what mime types need to be registered on the operating system.

Figure 1. isCOBOL file associations



Files with the extensions **.istc** and **.isws** are property files used to configure and guide isCOBOL Thin Client and isUPDATER applications respectively.

Here's an example of the contents of a file with the .istc extension:

```
iscobol.hostname=192.168.0.200
iscobol.port=10999
iscobol.default_program=MAIN_PROG
```

Double-clicking (or executing) this **.istc** file is the equivalent to running the following command:

```
isclient -hostname 192.168.0.200 -port 10999 MAIN_PROG
```

This command will run the program named MAIN_PROG on the isCOBOL Server running on port 10999 at the IP address 192.168.0.200.

Here's an example of the contents of a file with the .isws extension called myapp.isws:

```
swupdater.site=http://192.168.0.200:10996
swupdater.version.iscobol=975
swupdater.directory.iscobol=C:/Users/Veryant/isCOBOL THIN2019R1/lib
swupdater.directory.clean.iscobol=true
swupdater.version.iscobolNative=975
swupdater.directory.iscobolNative=C:/Users/Veryant/isCOBOL THIN2019R1/bin
swupdater.directory.clean.iscobolNative=true
swupdater.version.myApp=0
swupdater.directory.myApp=C:/myApp
swupdater.directory.clean.myApp=true
swupdater.mainclass=com.iscobol.invoke.Isrun -c C:/app/app.properties MYPROG
```

Double-clicking (or executing this **.isws** file is the equivalent to running the following command:

```
isupdater -c myapp.isws
```

The command above directs the isCOBOL updater tool to download the updated resources from the isCOBOL HTTP Server running on port 10996 of IP address 192.168.0.200 if necessary. When the download is complete, the *com.iscobol.invoke.Isrun* class is executed to run the application.

Responsive Layout Manager

Responsive design is an approach to develop user interfaces that render well on a variety of devices with different window and screen sizes. The viewer proximity is also considered as part of the viewing context. Content, design, and performance are factored across all devices to ensure usability and satisfaction.

Content is like water, "You put water into a cup, it becomes the cup. You put water into the bottle it becomes the bottle, you put water into the barrel, it becomes the barrel…"

Starting with isCOBOL 2019R1 screen sections can now be responsive, allowing the controls to be resized, moved or hidden based on the window's horizontal size when running in stand-alone, Thin Client and WebClient environments. This allows the user interface to adapt to multiple devices, screen sizes, and screen resolutions. It also allows the user to resize an application window or rotate a screen, a required behavior in the era of mobile devices and mobile-first development.

To enable a responsive layout, a new layout manager named **LM-RESPONSIVE** has been added to the DISPLAY WINDOW and DISPLAY TOOL-BAR statements, and the **LAYOUT-DATA** property on controls can define behavior based on window size. An **LM-RESPONSIVE** layout includes all the same rules provided with **LM-SCALE** layout.

In a Responsive Layout Manager design, you must define **media queries** to create **sensible breakpoints** for layouts and interfaces. These breakpoints are mostly based on minimum viewport widths and allow you to scale up elements as the view changes.

The responsive breakpoints are defined in the LM-RESPONSIVE handle of the layout manager.

The following code snippet defines four sensible breakpoints; "smartphone" with a screen size up to 799 pixels, "tablet" from 800 pixels to 1023 pixels, "desktop" from 1024 pixels to 1599 pixels and "widemonitor" with more than 1600 pixels.

```
77 responsive-layout handle of layout-manager, lm-responsive
   "smartphone=1 pixels, tablet=800 pixels, " &
   "desktop=1024 pixels, widemonitor=1600 pixels".
```

COBOL developers can also define **sensible breakpoints** using a CELL (or CELLS) unit approach. The following code snippet shows how to use CELLS in the handle of a layout manager declared in WORKING-STORAGE SECTION.

```
77  responsive-layout handle of layout-manager, lm-responsive
    "xsmall=1 cells, small=14 cells, " &
    "medium=40 cells, large=69 cells".
```

The name of these **sensible breakpoints** must be used in the SCREEN SECTION to define how LAYOUT-DATA works for each UI control. For example, in the following entry-field the LAYOUT-DATA defines:

- LINE, COL and SIZE properties for the “small” breakpoint are 3.5, 2, and 14 cells respectively, replacing the default values of 2, 1.3, and 54 cells.
- LINE, COL and SIZE properties for the “medium” breakpoint are 3.5, 2, and 38 cells respectively, replacing default values of 2, 1.3, and 54 cells.
- RESIZE-X property is used for “small” and “medium” breakpoints, replacing default LM-SCALE behavior (RESIZE-X-ANY + MOVE-BOTH-ANY)

```
03 EF-TITLE
   entry-field
   line 2 lines 1.3 col 12 size 54 cells
   layout-data  "line-small 3.5 cells " &
                "col-small 2 cells " &
                "size-small 14 cells " &
                "resize-x-small " &
                "line-medium 3.5 cells " &
                "col-medium 2 cells " &
                "size-medium 38 cells " &
                "resize-x-medium".
```

The Responsive Layout manager allows you to specify some directives in LAYOUT-DATA that rule how to make UI components visible or hidden.

For example, in the following push-buttons the LAYOUT-DATA specifies that:

- PB-EXIT is hidden for the “small” breakpoint and visible on all other breakpoints
- PB-MENU is visible for the “small” breakpoint and hidden for all other breakpoints
- NO-SCALE clause replaces the default LM-SCALE behavior MOVE-BOTH-ANY

```
03 PB-EXIT push-button self-act
    line 2 col 2 lines 16 size 10 cells
    title "Exit"
    ...
    layout-data          "hidden-small " &
                         "no-scale".
 ...
03 PB-MENU push-button self-act
    title "menu"
    line 2 col 2 lines 16 size 16
    ...
    layout-data          "visible-small " &
                         "no-scale"
```

To enable responsive layout, the DISPLAY WINDOW and DISPLAY TOOL-BAR statements also need to use the LAYOUT-MANAGER property:

```
display standard graphical window resizable
         layout-manager responsive-layout

display tool-bar moveable
         layout-manager responsive-layout
```

Figure 2, *"small" breakpoint,* shows how the program runs under small devices according to all "small" breakpoint rules defined in LAYOUT-DATA property with the typical "hamburger" menu.

Figure 2. "small" breakpoint

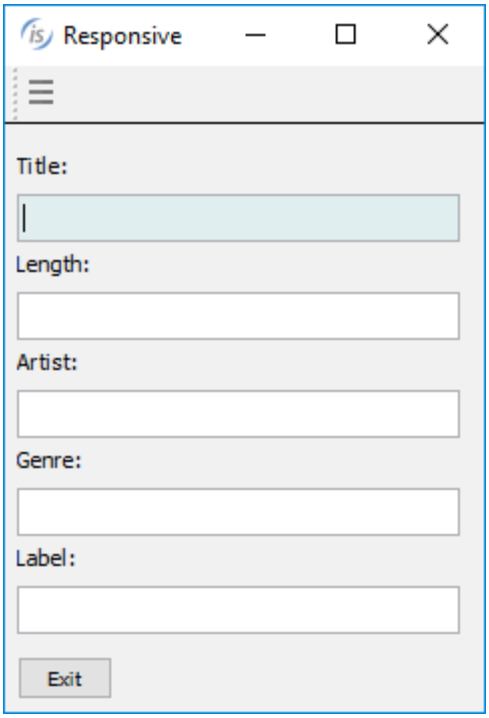


Figure 3, *"medium" breakpoint*, shows how the program runs under medium devices. Now the toolbar contains new buttons (visible on "medium" breakpoint) and the "hamburger" button menu is hidden.

Figure 3. "medium" breakpoint

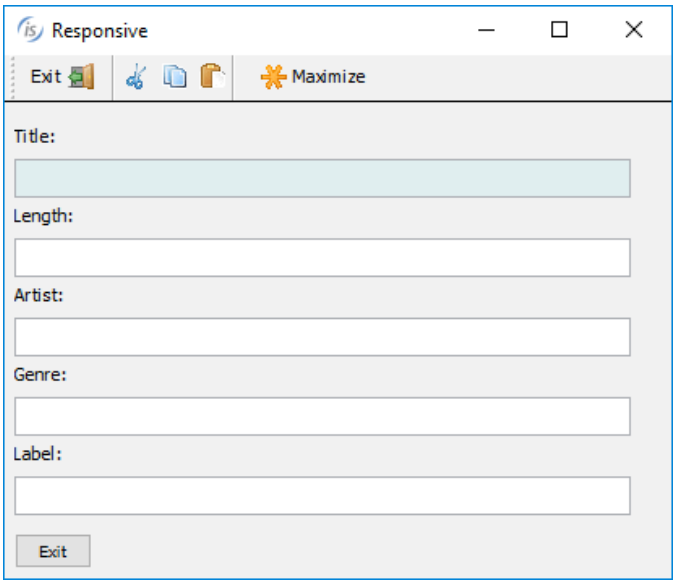
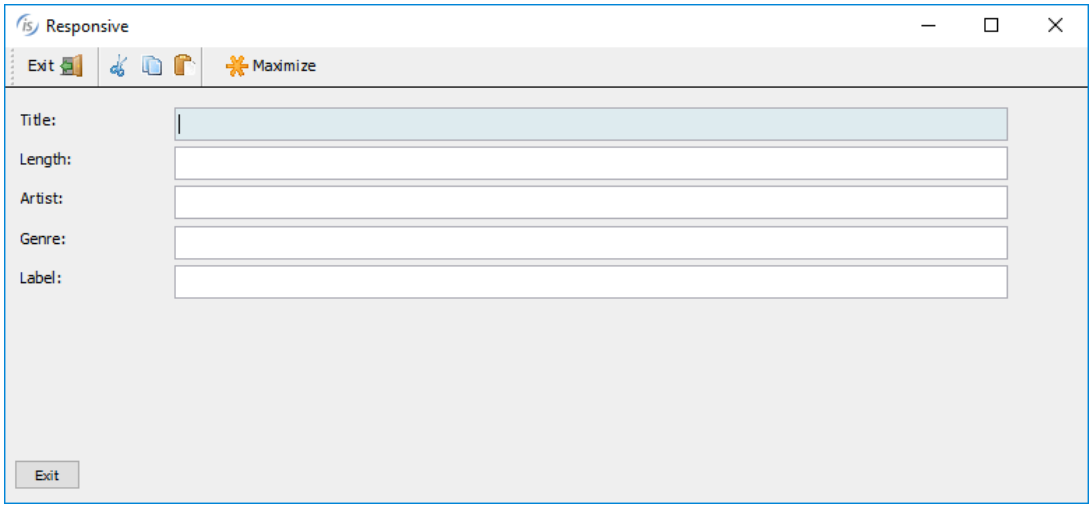


Figure 4, *"large" breakpoint*, shows the same program running on a large device, with space enough to have a label and an entry-field on the same line

Figure 4. "large" breakpoint



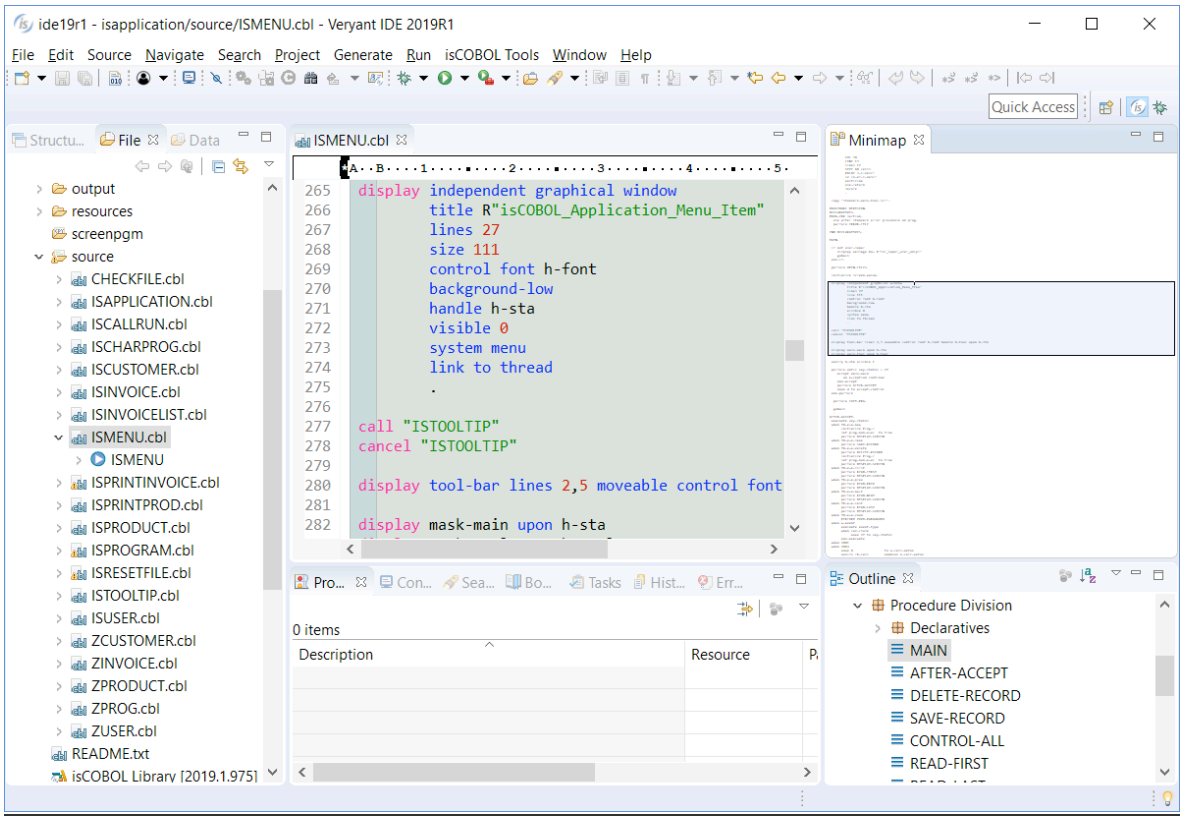
isCOBOL IDE Enhancements

The isCOBOL 2019 R1 IDE is now built on the latest Eclipse 2018-09. This release is the first quarterly Eclipse Simultaneous Release of Eclipse 4.9 that provides full support of Oracle JDK 11 and OpenJDK 11.

Eclipse features

The newest release of isCOBOL IDE encompasses all the new features in Eclipse 2018-09 and all new features in Eclipse Photon. One of those new features is the Editor Minimap, which gives developers a high-level overview of the content of the currently active text editor. This aids in navigation and gives you a better understanding of your code, as shown in Figure 5, *Editor minimap*.

Figure 5. Editor minimap



Default Debug perspective layout changed. The aim is to give the editor area more space and to show more relevant information without scrolling. Display view, Expressions view, and Project Explorer are now shown by default, Problems view replaces Tasks.

COBOL-WOW support

The isCOBOL 2019 R1 IDE can import existing COBOL-WOW projects, providing screen designers and code editors to ease maintenance and development. When imported in the IDE, developers have access to Eclipse's advanced editor and tools.

All of COBOL-WOW GUI widgets are supported and written in Java for 100% portability across environments, with an updated and modern look.

isCOBOL IDE support for COBOL-WOW provides the following UI widget:

Command Button, Check Box, Option Button, Static Test, List Box, Combo Box, Vertical, Scroll, Horizontal Scroll, Toolbar, Timer, Month Calendar, Rounded Rectangle, Eclipse Edit Box, Group Box, Bitmap, Animation Control, Progress Bar, Track Bar, Status Bar, Up/Down Control, Tab Control, Data Time Picker, Line, Rectangle.

Also, most of COBOL-WOW routines are provided in order to have a seamless execution of generated COBOL source code:

Axbindeventarguments, Axdomethod, Axunbindeventarguments, Checkmenuitem, Closewindow, Deletemenu, Drawmenubar, Enablemenuitem Enablewindow, Findwindow, Getactivewindow, Getcursorpos, Getenvironmentvariable, Getfocus, Getmenu, Getsubmenu, Getwindowsdirectory, Ischild, Iswindow, Messagebeep, Messagebox, Modifymenu, Openicon, Sendmessage, Setactivewindow, Setfocus, Showwindow, Winexec, Wowadditem, Wowclear, Wowclearwaitcursor Wowcreatewindow,Wowdestroywindow, Wowdiscardevents, Wowgetfocus, Wowgetindexprop, Wowgetmessage, Wowgetnum, Wowgetprop, Wowinitallcontrols, Wowinitcontrol, Wowmessagebox, Wowmove, Wowmulticontrolgetprop, Wowmulticontrolsetprop,Wowpeekmessag, Wowrefresh, Wowremoveitem, Wowresetwaitcursor, Wowsetfocus, Wowsetindexprop, Wowsetnextctrl, Wowsetnum, Wowsetprevctrl, Wowsetprop, Wowsetstriptrailing, Wowsetwaitcursor, Wowversion1

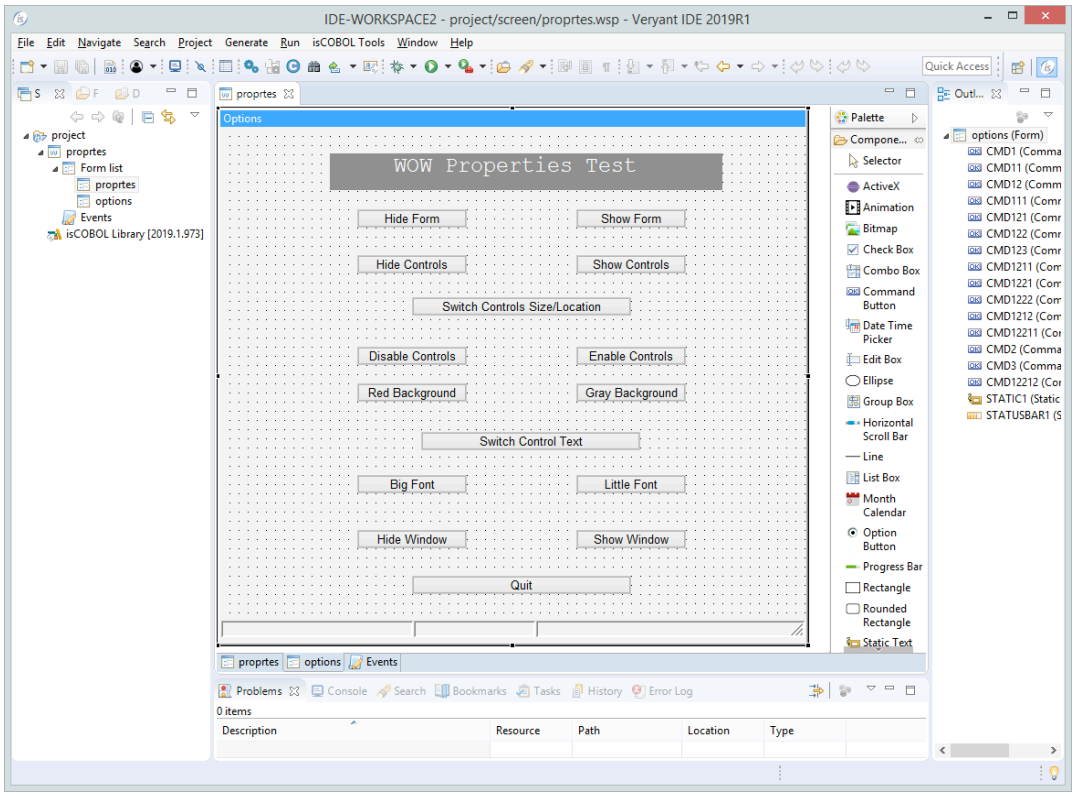
Once your code is in the isCOBOL Evolve environment the possibilities are endless, from object-oriented COBOL programming to all the benefits of working within the Java ecosystem. This includes the close-to-infinite number of libraries and toolkits you can use to tackle every conceivable task as well as the ability to deploy with Veryant's Application Server and Thin Client or Web Client technologies.

COBOL-WOW GUI and isCOBOL GUI can be used together, as long as they are in separate programs, giving you greater flexibility. Almost all of the WOW library routines to manage GUI widgets are provided by the isCOBOL Runtime. Developers can choose to implement new requirements using either WOW programming or the SCREEN SECTION approach.

COBOL-WOW programs converted to isCOBOL can also run in Thin client or Web client mode in an Application Server environment, allowing you to leverage all of Veryant's solutions as well as platform independence without any changes.

Figure 6, *COBOL WOW Support*, shows a COBOL WOW program imported in isCOBOL IDE. You can see the WYSWYG GUI Painter, the list of supported GUI widgets, the event editor and more.

Figure 6. COBOL WOW Support

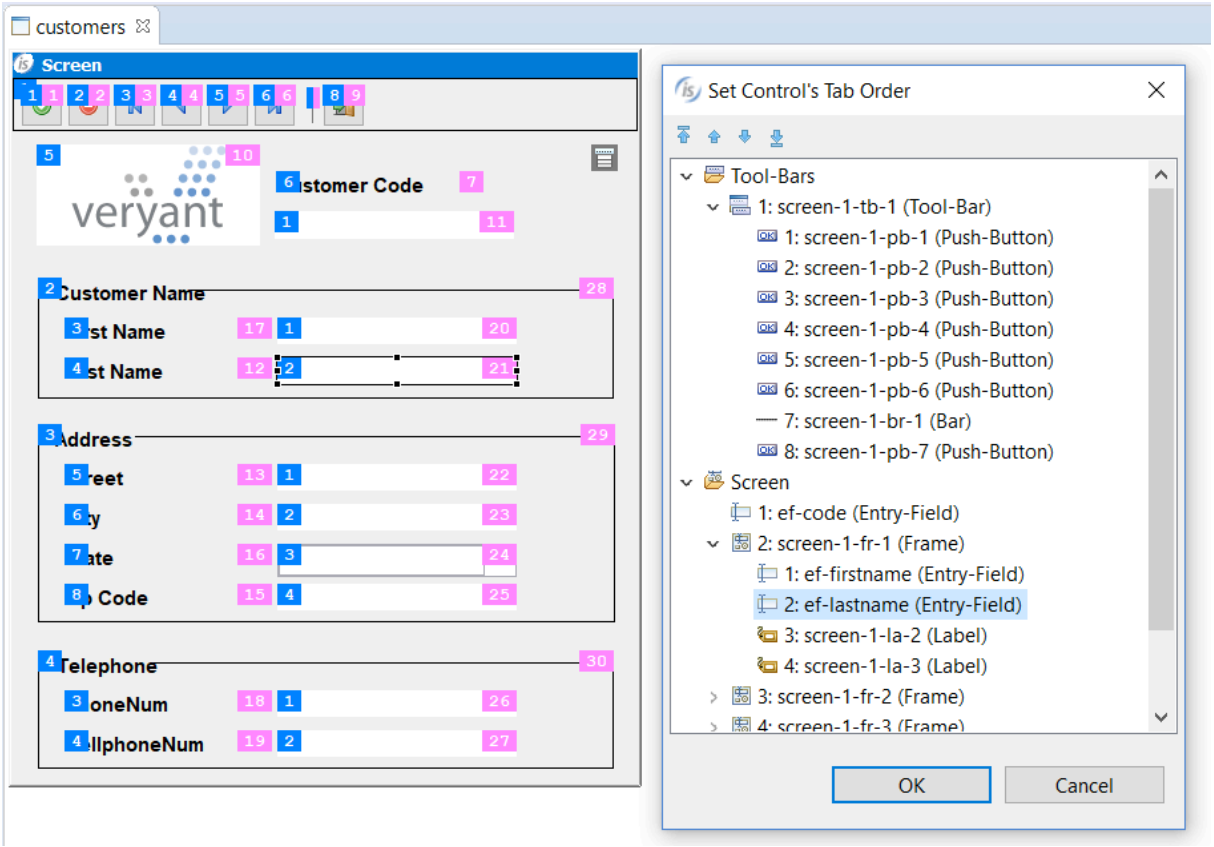


New screen painter features

The isCOBOL screen painter can now display tabulation order and control IDs of controls, as shown in Figure 7, *Tab order and control ID.* The tab order of controls is displayed on the left side, while control ID is displayed on the right side of the control.

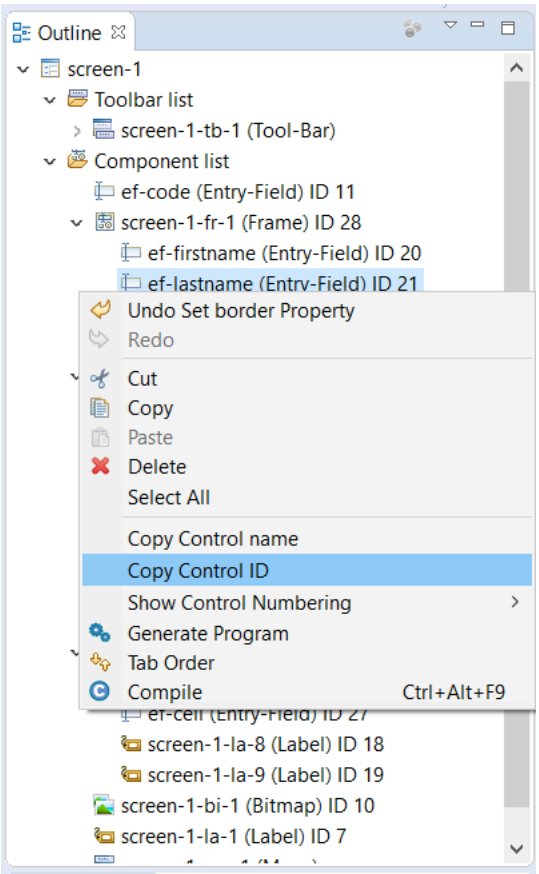
The Set Control's Tab Order property window now supports drag and drop of controls to visually set tabulation order.

Figure 7. Tab order and Control ID



The Outline view in screen painter has been enhanced to show the controls' ID, and the context menu of the control now has "Copy Control name" and "Copy Control ID" options to copy names and IDs to the system clipboard, as depicted in Figure 8. *Outline view*

Figure 8. Outline view



New IDE preferences can be used to customize foreground and background colors for the Tab Order and Control ID features of the Screen Designer. The new "Generate linked files for copy books not belonging to the workspace" preference can be used to control automatic generation of link files in the copy folder of the project for each copybook specified in the source files. When this option is cleared no link file will be generated, which can be useful to avoid cluttering the copy folder on large projects. The copybooks can still be opened from the source code editor, by clicking the triangle icon in the COPY statement.

Framework improvements

Performance of CALL statements has been greatly improved, with up to 55% better performance compared to the previous version. New configuration properties and new library routines have been introduced.

Performance of CALL statements

Performance of CALL statements have been improved across the board, especially when using CALL/CANCEL statements under `code_prefix`, the feature that allows hot updates of running programs. Code prefix proves especially useful in server environments, for example when running under isCOBOL Server.

Refining class loading, and allowing developers to control it programmatically has achieved these improvements. When using code_prefix in previous versions, the first call and each subsequent call to the program after a cancel would run a check on the file system to see if the file had been updated since the last call. If so, the runtime unloaded the old version of the program and loaded the new. The check can be time consuming if executed frequently. Developers can now use the new `iscobol.code_prefix.reload=false` property to disable automatic reloading and leverage the new library routine **C$UNLOAD** to unload the program manually and allow updating.

Another improvement concerns remote calls configured using the `remote.code_prefix` property, which leverages more optimized runtime TCP communications, and does not need source code modification to enable performance gains in calls using code_prefix or the classpath.

A table of performance gains is shown in Figure 9, *Performance improvements,* and shows a performance comparison between isCOBOL 2018R2 and isCOBOL 2019R1. The tests were run in Windows 10 64-bit on an Intel Core i5 Processor 4440+ clocked at 3.10 GHz with 8GB of RAM, using Oracle JDK1.8.0_192. All times are in seconds.

The called program contains a linkage group data item with 50 child items. The number of CALL/CANCEL iterations is 10,000.

The code_prefix tests under 2019R1 are in conjunction with code_prefix.reload=false

Figure 9. Performance improvements

CALL statement type	configuration	2018 R2	2019 R1	% improvement
CALL	classpath	0,02	0,02	0,00%
	code_prefix	0,02	0,02	0,00%
CALL / CANCEL	classpath	1,34	1,33	0,75%
	code_prefix	**2,48**	**1,12**	**54,84%**
Remote CALL	classpath	10,55	8,14	22,84%
	code_prefix	10,77	7,96	26,09%
Remote CALL / CANCEL	classpath	26,84	24,10	10,21%
	code_prefix	30,18	22,87	24,22%
CALL Client	classpath	10,21	7,79	23,70%
	code_prefix	10,36	7,22	30,31%
CALL Client / CANCEL Client	classpath	16,25	13,33	17,97%
	code_prefix	17,16	13,14	23,43%
Total		136,18	107,04	21,40%

New configuration properties:

New configuration properties have been introduced in the Framework:

`iscobol.code_prefix.reload=false`

to disable the automatic class reload, as discussed in the performance improvement section of this document.

New configuration properties are used for starting up programs with .istc configuration files:

- `iscobol.default_program=PGM` to specify the main program to execute when it's not passed

  `iscobol.default_options=options` to specify the options for isCOBOL execution (standalone or thin client)

Additional configuration properties:

- `iscobol.call_run.sync=true` to execute the CALL RUN synchronously instead of asynchronously
- `iscobol.esql.indicator_trunc_on_call=false` to set the indicator variable to 0 when the stored procedure's output parameter value doesn't fit the host variable.
- `iscobol.file.indd=myindd` to associate custom file handlers to the files specified by INDD directive
- `iscobol.file.outdd=myoutdd` to associate custom file handlers to the files specified by OUTDD directive
- `iscobol.gui.entryfield.notify_change_delay=n` to affect all entry-fields
- `iscobol.key.default_shortcuts_enabled=true` to intercept shortcuts like Ctrl+C
- `iscobol.upper_lower_method=n` default value 1, where n can be:
    - 1, to use the method of String.toUpperCase / toLowerCase
    - 2, to use the method of Character.toUpperCase / toLowerCase

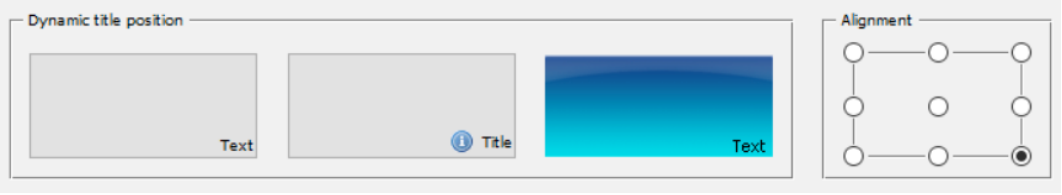
Enhanced push button title position

Push buttons have been enhanced to allow finer control of title placement when a large bitmap is assigned to the button. A title can now be positioned relative to the button bitmap when a bitmap fully covers the button surface.

Figure 10, *Title position,* shows the result of the following code snippet which applies the new position to the third button:

```
modify pb-3 title-position TITLE_OVERLAPPED_BITMAP_BOTTOM_RIGHT
```

Figure 10. Title position



New library routines

The new **C$UNLOAD** routine can be used to unload programs loaded in memory without restarting the application. In order to take advantage of this new feature, you need to configure the application to be executed with "code_prefix" class loader as well as set the following configuration property:

```
iscobol.code_prefix.reload=false
```

Two new library routines have been implemented to delete or rename C-TREE files on the server side.:

- **C$FSDELETE** to delete indexed files using the File Manager's internal API
- **C$FSRENAME** to rename indexed files using the File Manager's internal API

Miscellaneous routines added in the newest release of isCOBOL 2019 R1 include the following:

CBL_EXEC_RUN_UNIT can be used to run a unit that inherits the environment variables of the calling program

C$ENCRYPT and **C$DECRYPT** can encrypt and decrypt a file using several cryptographic algorithms, as configured in the `iscobol.crypt.algorithm` property, such as:

- AES Advanced Encryption Standard as specified by NIST in FIPS 197.
- AESWrap The AES key wrapping algorithm as described in RFC 3394.
- ARCFOUR A stream cipher believed to be fully interoperable with the RC4 cipher
- Blowfish The Blowfish block cipher designed by Bruce Schneier.
- CCM Counter/CBC Mode, as defined in NIST Special Publication SP 800-38C.
- DES The Digital Encryption Standard as described in FIPS PUB 46-3.
- DESede Triple DES Encryption (also known as DES-EDE, 3DES, or Triple-DES)
- ECIES Elliptic Curve Integrated Encryption Scheme
- GCM Galois/Counter Mode, as defined in NIST Special Publication
- RC2 Variable-key-size encryption algorithms developed by Ron Rivest
- RC4 Variable-key-size encryption algorithms developed by Ron Rivest
- RC5 Variable-key-size encryption algorithms developed by Ron Rivest
- RSA encryption algorithm as defined in PKCS #1

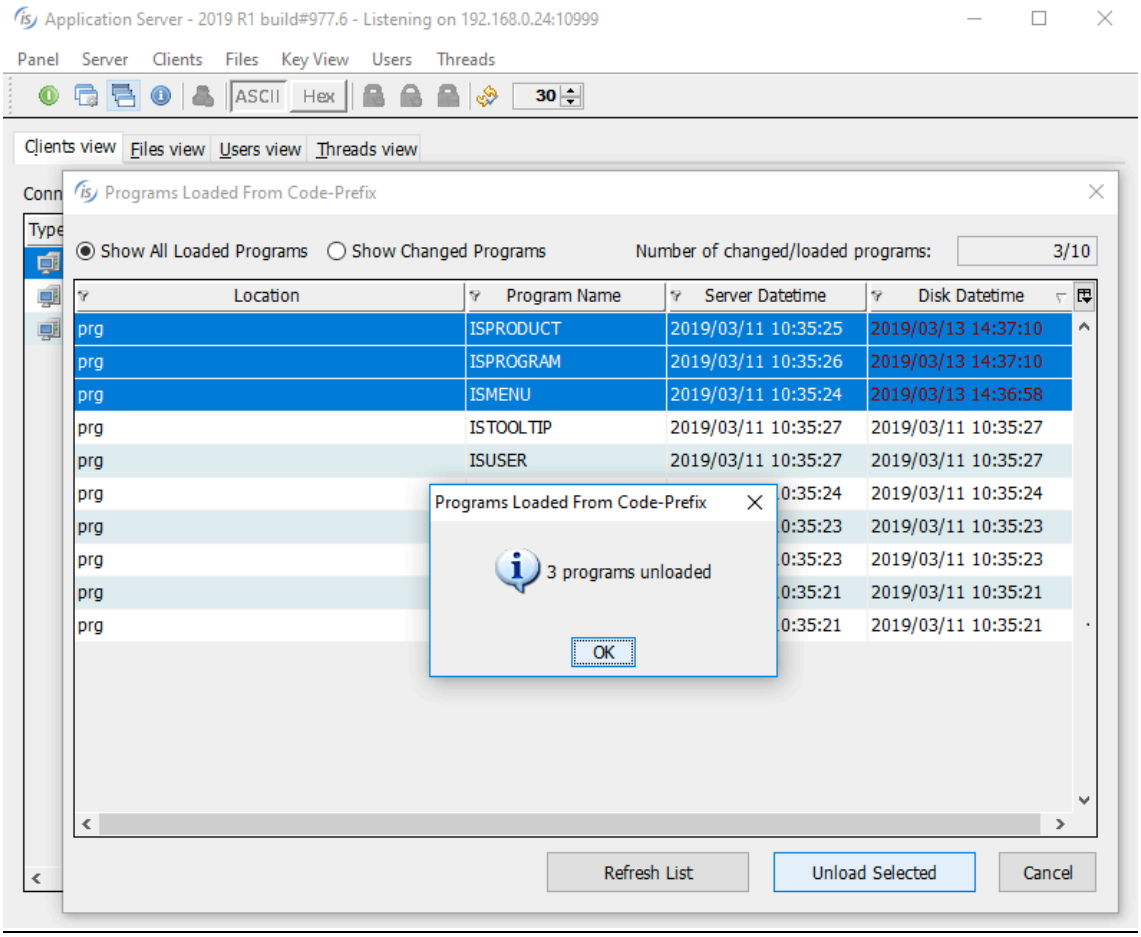
The library routine **C$FORNAME** can be used to check if a class is available on the system .

isCOBOL Server

As depicted in Figure 11, *isCOBOL Server Panel*, the isCOBOL Server Panel now provides a "Programs Loaded From Code-Prefix" dialog that shows programs loaded under `iscobol.code_prefix`, the timestamp of the loaded programs and of corresponding disk class file, and allows unloading of selected COBOL programs without restarting the application server. This feature allows developers to provide new versions of COBOL programs to be loaded to isCOBOL Server's users for immediate use.

To take advantage of the unload feature, you need to configure the application to be executed with "code_prefix" class loader having `iscobol.code_prefix.reload` set to false.

Figure 11. isCOBOL Server Panel



isCOBOL Compiler

Starting from isCOBOL 2019R1, OOP syntax has been enhanced to support the ANSI SO/IEC 1989:2014's INTERFACE-ID paragraph.

The INTERFACE-ID paragraph indicates that this identification division is introducing an interface definition, specifying the name that identifies the interface and assigning interface attributes to the interface.

As an extension from ANSI SO/IEC 1989:2014, the isCOBOL 2019R1 Compiler also provides optional DEFAULT methods and auto-boxing of Java primitive data types.

New compiler options and new syntax have been added to enhance compatibility with other COBOL dialects.

New OOP Syntax

The following program defines a new interface, and defines a default method:

```
identification division.
INTERFACE-ID. myInterface as "myInterface".
identification division.
object.
procedure division.
identification division.
method-id. metInterface as "metInterface" DEFAULT.
procedure division.
main.
  display "metInterface"
  goback.
end method.
end object.
END INTERFACE.
```

The following code defines a class that implements the previous interface:

```
identification division.
class-id. myClass as "myClass" implements myInterface.
configuration section.
repository.
   class myInterface as "myInterface".
identification division.
object.
procedure division.
identification division.
method-id. metClass as "metClass".
procedure division.
   display "metClass"
   goback.
end method.
end object.
```

The following program defines an object variable and instantiates it, then calls its default method:

```
PROGRAM-ID. PROG.
CONFIGURATION SECTION.
REPOSITORY.
   CLASS myClass AS "myClass".
WORKING-STORAGE SECTION.
77 OBJ OBJECT REFERENCE myClass.
PROCEDURE DIVISION.
MAIN.
   set OBJ = myClass:>new().
   OBJ:>metInterface().
   OBJ:>metClass.
GOBACK.
```

Auto boxing on primitive types

Primitive types can now be used as if they were COBOL variables, making source code more readable and flexible:

```
PROGRAM-ID. AUTO-BOXING.
WORKING-STORAGE SECTION.
77 var1-t object reference "boolean".
PROCEDURE DIVISION.
MAIN.
set var1-t to true
if var1-t
      display "true"
else
      display "false"
end-if
GOBACK.
```

New Compiler options

-vansi can be used to implicitly add FROM/UPON CONSOLE phrases to ACCEPT and DISPLAY statements. With this option the code:

```
DISPLAY "ABC" line 2 col 2 reverse
ACCEPT VARX line 3 col 2 underline
```

will be translated as:

```
DISPLAY "ABC" UPON CONSOLE
ACCEPT VARX FROM CONSOLE
```

Other compiler options added in isCOBOL 2019 R1:

- -dvext=nn to initialize external items to a default byte value
- -dvexta to initialize external items to a default byte value, for compatibility with ACUCOBOL-GT®
- -dznt for compatibility with MicroFocus® NOTRUNC flag
- -dzta for compatibility with MicroFocus® TRUNC"ANSI" flag
- -sl to allow AREA B to extend to the end of the line, regardless of line length. The same format can also be achieved on a single source file by using the ">>IMP MARGIN-R IS AFTER END OF RECORD" compiler directive.

New Syntax for compatibility

The following INSPECT statement clauses are now supported for compatibility with GnuCOBOL (formerly OpenCOBOL).

```
INSPECT VARX TALLYING/REPLACING VAR9 FOR TRAILING "a" BEFORE "B"

INSPECT VARX REPLACING TRAILING "A" BY "B" BEFORE "C"
```

Background and foreground colors can now be specified by name instead of by number, improving compatibility with RM/COBOL.

The code below is now supported:

```
DISPLAY "Text on Line 1" line 1 col 1 background BLUE
                                      foreground RED
```

isCOBOL EIS

isCOBOL EIS provides a utility called Service Bridge to automatically generate SOAP and REST services from a legacy COBOL program with Linkage Section.

Starting from isCOBOL2019R1, custom COBOL code can now be inserted into the source code that is automatically generated by the Web Service Bridge feature.

The Native Boolean data type is now supported in JSON and XML data definitions.

Custom code

Custom code can be inserted into the source code that is automatically generated by the Web Service Bridge feature. The generated code is embedded in the `*>start` and `*>end` directives. Code that is written outside those tags will be preserved during the automatic generation of the tagged areas, allowing custom behavior to be included as part of the program. The code below shows how to add a CALL to MYPROG after input parameters have been parsed by the runtime and before the ws-info legacy program is invoked.

```
*>start {iscobol}http-to-linkage
    move  PAR1-in to intPAR1;;
*>end {iscobol}http-to-linkage

*> This is my custom code written outside the Tagged Areas
   CALL "MYPROG" USING intPAR1

*>start {iscobol}call
    call "ws-info" using
                   intPAR1
*>end {iscobol}call
```

A typical use case of custom code is to check for security and authentication tokens passed in the HTTP header of the request, which would normally be ignored by the Service Bridge feature. Logging and transformation of input and output parameters are other use cases for custom code.

Boolean data type

Boolean type variables can now be specified in the request and response elements that will be transformed in JSON or XML data streams.

The sample code below shows how to use the new `IS BOOLEAN` syntax to specify that a variable will receive true/false values. If the variable is defined as pic 9, the value 1 will be rendered as “true” and 0 as “false” when writing, and will be parsed as value 1 when true is read and 0 when false is read. When pic x is specified, the resulting data will be the string “true” or “false”.

```
01  js-stream-data identified by "names".
 02 identified by "list"    occurs dynamic capacity js-cap.
    04 js-first-name          identified by "FirstName".
       06 js-first-name-data  pic x any length.
    04 js-last-name           identified by "LastName".
       06 js-last-name-data   pic x any length.
    04 js-city                identified by "City".
       06 js-city-data        pic x any length.
    04 js-foreign             identified by "Foreign"
                              IS BOOLEAN.
       06 js-foreign-data     pic x any length.
```

The JSON generated:

```
{
    "names":{
       "list":[
         {
            "FirstName":"John",
            "LastName":"Red",
            "City":"Miami",
            "Foreign":false
         },
         {
           "FirstName":"Mario",
           "LastName":"Rossi",
           "City":"Milan",
           "Foreign":true
         }
     ]
    }
}
```

isCOBOL Database Bridge

isCOBOL Database Bridge, the seamless RDBMS Interface for indexed COBOL Files, now also support binary sequential, line sequential and relative files.

This new feature allows you to take advantage of all the features provided through the RDBMS; for example online backup, encryption, users managements rights etc.

As default. isCOBOL Database Bridge generates just indexed COBOL files. A new compilation property has been added to also enable sequential and relative file generation:

```
iscobol.compiler.easydb.index_only=false
```

The edbiis standalone command can now process any XML file created by the compiler using –efa option.

To run a COBOL program that needs to map sequential and relative COBOL files into an RDBMS, you should set the following properties:

```
iscobol.file.sequential=easydb
iscobol.file.linesequential=easydb
iscobol.file.relative=easydb
```