veryant

isCOBOL^{TM} Evolve

isCOBOL Evolve 2024 Release 2 Overview

isCOBOL Evolve 2024 Release 2 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2024 R2.

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

The new incremental compiler feature is available to speed up compilations for projects that contain a large number of source files. The compiler supports syntax for Boolean and USAGE-BIT.

The updated isCOBOL Debugger can now fully access the variables of programs in the stack.

isCOBOL Evolve 2024 R2 introduces a new GUI control named CHIPS-BOX as well as other graphical enhancements.

Details on these enhancements and updates are included below.

IsCOBOL Compiler

IsCOBOL Compiler 2024 R2 supports incremental compilation through new options as well as implementing COBOL Boolean and USAGE BIT syntax. It also improves the OOP (Object Oriented Programming) syntax by allowing the VALUE clause in object reference data items.

Incremental compilation

The Incremental Compiler is a compilation strategy in which only programs with modified text, COPY files, or INCLUDE files get recompiled. The changes will be merged with previously compiled code to form new Java code. When you compile your entire source code base using this new feature, the incremental compiler will result in a faster compilation step. The main advantage of this strategy is the performance boost with massive compilations; for example, when using CI (Continuous Integration) tools where source code in often recompiled to keep classes up to date with changes made in source code during development. Until the previous release a complete recompilation was done running a command such as:

```
iscc -options *.cbl
```

To enable the incremental compilation, use the new -incr option passing the value "build", such as:

```
iscc -options -incr=build *.cbl
```

The build operation uses the incr.iscc file contained in the current folder to check if a source file needs to be recompiled because of a change after the last successful compilation. The compiler checks the timestamp of the source file and compares it to the timestamp of the .class file.

The incr.iscc file name and location can be customized by passing an alternative pathname after the build and clean options with a semicolon separator, for example:

```
iscc -options -incr=build;..\resources\CompInfo folder1\*.cbl
```

uses CompInfo from ..\resources instead of incr.iscc from the current directory.

Only one alternative pathname can be supplied.

The use of a customized pathname allows you also to reuse the same file for different compilations, such as

```
cd source1
iscc -options -incr=build;..\resources\CompInfo folder1\*.cbl
cd ..\source2
iscc -options -incr=build;..\resources\CompInfo folder2\*.cbl
```

To force full recompilation, the clean parameter can be passed to the -incr option, for example:

```
iscc -options -incr=clean,build *.cbl
```

The values passed to the -incr option are equivalent to the isCOBOL IDE Eclipse plugin, where you have the options to build the project or clean and then build. Now this feature is also available to developers. If you prefer to use a command-line approach and batch compilation typical for massive build operations, you can now take use incremental compilation.

COBOL Boolean and USAGE BIT support

A new type of PICTURE named 1 is supported to manage Boolean data items. Boolean variables are typically used for flag or a list of flags data items that can be set to true or false and then used in conditions in the program logic. To simplify migration of numeric variables to Boolean variables, two new functions are supported.

- INTEGER-OF-BOOLEAN to convert an integer value to a Boolean value
- BOOLEAN-OF-INTEGER to convert a Boolean value to an integer value

The PICTURE 1 also supports the USAGE BIT clause to manage the variable with bit series.

This is an example on using the Boolean data type:

```
...
01  bit-item    PIC 1(8) USAGE BIT.
01  num-item    PIC 9(5).
01  num-item-2  PIC 9(5).
...
MOVE 123 to num-item
MOVE FUNCTION BOOLEAN-OF-INTEGER(num-item, 8) TO bit-item.
display bit-item  | this shows 01111011
if bit-item(1:1)  | this is false
   display "bit 1 is true"
else
   display "bit 1 is false"
end-if
if bit-item(2:1) | this is true
   display "bit 2 is true"
else
   display "bit 2 is false"
end-if
...
COMPUTE num-item-2 = FUNCTION INTEGER-OF-BOOLEAN (bit-item).
display num-item-2 | this shows 123
```

Additional compiler options:

The additional compiler options implemented in this release are: -brand and -csqq2.

The -brand="value" option stores custom information in the class. This feature is useful when a piece of information needs to be stored in the .class file at compile time and can then be accessed by the running code.

For example, a code snippet like:

```
move function compiled-info() to w-compinfo
initialize w-count
inspect w-compinfo tallying w-count for all "-brand="
if w-count = 0
   display "Program compiled without -brand option"
else
   inspect w-compinfo tallying w-count for characters before "-brand="
   add 7 to w-count
   move w-compinfo(w-count:4) to w-myvers convert
end-if
if w-myvers > 132
   call "newprog" using w-params
end-if
```

will execute the CALL “newprog” only if the value passed to the -brand option is > 133, making the execution dependent on how the program has been compiled. Compiling with these commands:

```
iscc -brand=132 MYPROG.cbl
```

or:

```
iscc -brand=133 MYPROG.cbl
```

makes the difference. To check for the brand information in a class, run the following command:

```
iscrun -info MYPROG
```

This will return the information that has been stored along with all other compiler options:

```
...
Compile flags: -g -oe -brand=133 -b -cghv
...
```

The preprocessor can also get this information by using the configuration iscobol.compiler.custompreproc. This will allow for different preprocessing logic to be applied depending on the value set with the -brand option.

The -csqq2 option is used to switch double and single quotes in ESQL code. This option is useful in sources where the quotes have not been written in the ESQL standard ruleset. By default, double quotes are used for identifiers and single quotes are used for alphanumeric values. But in cases where the sources contain the opposite situation, for example:

```
EXEC SQL
   INSERT INTO 'MixedCase' ('CharCol', 'NumCol') VALUES ("abc", 1)
END-EXEC
```

Compiling with the new -csqq2 option will treat the code as:

```
EXEC SQL
   INSERT INTO "MixedCase" ("CharCol", "NumCol") VALUES ('abc', 1)
END-EXEC
```

The SQL code will be executed correctly and the standards for JDBC drivers.

VALUE clause in object reference

The VALUE clause is used in working-storage data items to initialize a value at the beginning of the program. Starting from 2024 R2, the VALUE clause is also supported for object reference, allowing initialization at program startup for this kind of data as well. Code like the following:

```
repository.
   class jstring  as "java.lang.String"
   class jint     as "java.lang.Integer"
...
77 j1    object reference jstring value jstring::new("Value1").
77 j2    object reference jstring value "Value2".
77 jint1 object reference jint    value jint:>new(123).
77 jint2 object reference jint    value 456.
```

Will instantiate the 4 objects already with the provided values. Basically, it is the same as executing the following SET statements at the beginning of the procedure division:

```
set j1    = jstring::new("Value1")
set j2    = "Value2"
set jint1 = jint:>new(123)
set jint2 = 456
```

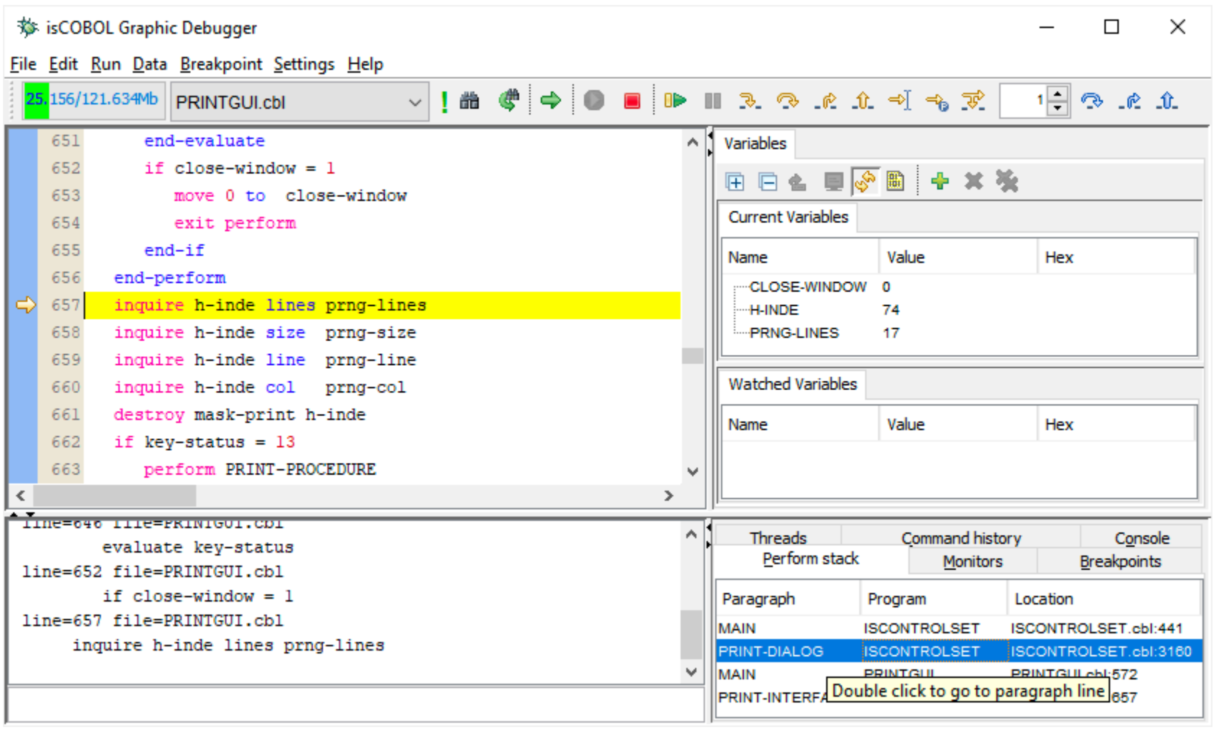
IsCOBOL Debugger

isCOBOL Debugger 2024 R2 supports full access to data items of programs and classes in the stack, granting complete control for developers. 2024R2 also introduces other enhancements to Perform stack view and Breakpoints.

Access to data items in the stack

During the execution of application using the debugger it's typical to access data items of the current program to show or change the values. However sometimes it's also necessary and useful for developers to access the values of data items defined in previous programs. For example, it can be useful to inspect variables in a caller program to check how parameters passed to the called program were assigned. Starting from 2024R2, isCOBOL Debugger supports access to data items contained in programs and classes in the stack. By double clicking on the needed line in the Perform stack view, as shown in Figure 1, *Open a program from the Stack*, the corresponding source code is loaded, and all the commands related to the data items are supported. When opening the dialogs related to data items, the new field Method/Program is already set to the appropriate source file name, as shown in Figure 2, *Quick Watch with Program set*.

Figure 1. Open a program from the Stack



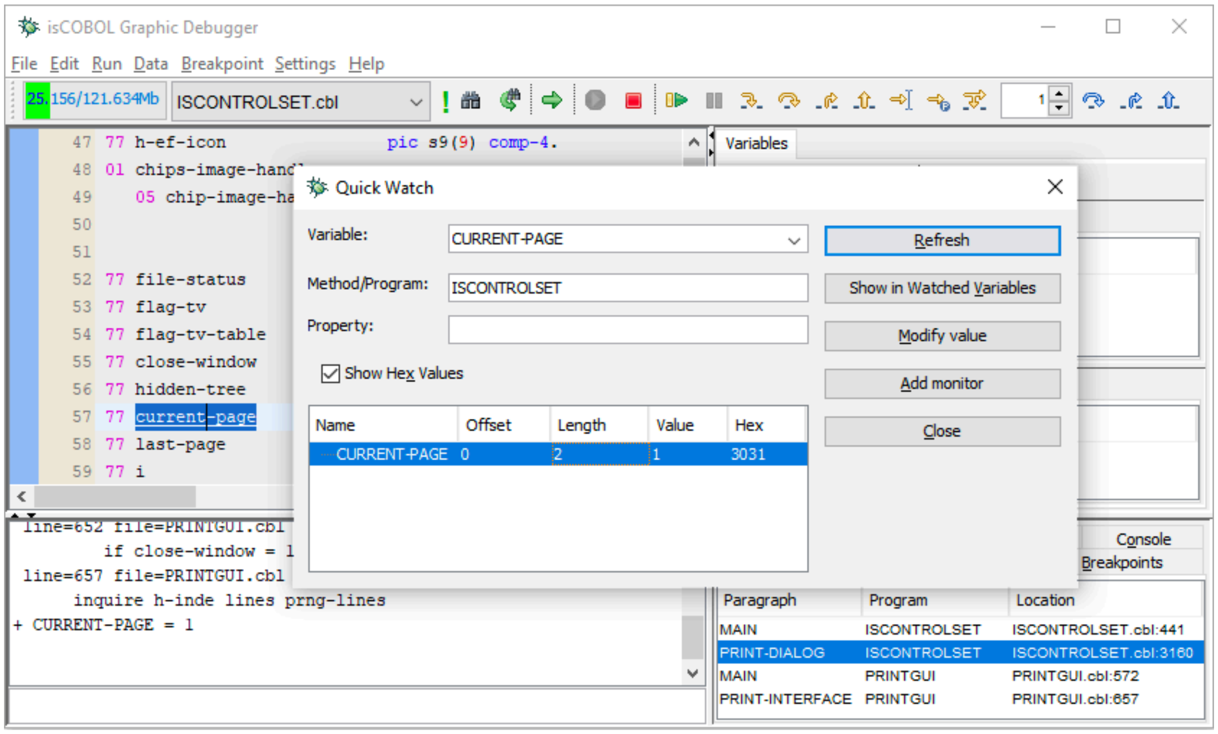
```
          end-evaluate
          if close-window = 1
             move 0 to  close-window
             exit perform
          end-if
       end-perform
       inquire h-inde lines prng-lines
       inquire h-inde size  prng-size
       inquire h-inde line  prng-line
       inquire h-inde col   prng-col
       destroy mask-print h-inde
       if key-status = 13
          perform PRINT-PROCEDURE
```

Name	Value	Hex
CLOSE-WINDOW	0	
H-INDE	74	
PRNG-LINES	17	

```
line=646 file=PRINTGUI.cbl
        evaluate key-status
line=652 file=PRINTGUI.cbl
        if close-window = 1
line=657 file=PRINTGUI.cbl
     inquire h-inde lines prng-lines
```

Paragraph	Program	Location
MAIN	ISCONTROLSET	ISCONTROLSET.cbl:441
PRINT-DIALOG	ISCONTROLSET	ISCONTROLSET.cbl:3160
MAIN	PRINTGUI	PRINTGUI.cbl:572
PRINT-INTERFA		657

Figure 2. Quick Watch with Program set.



```
 77 h-ef-icon                   pic s9(9) comp-4.
 01 chips-image-hand
    05 chip-image-ha


 77 file-status
 77 flag-tv
 77 flag-tv-table
 77 close-window
 77 hidden-tree
 77 current-page
 77 last-page
 77 i
```

Variable: CURRENT-PAGE
Method/Program: ISCONTROLSET
Property:
[✓] Show Hex Values

Name	Offset	Length	Value	Hex
CURRENT-PAGE	0	2	1	3031

```
line=652 file=PRINTGUI.cbl
        if close-window = 1
line=657 file=PRINTGUI.cbl
     inquire h-inde lines prng-lines
+ CURRENT-PAGE = 1
```

Paragraph	Program	Location
MAIN	ISCONTROLSET	ISCONTROLSET.cbl:441
PRINT-DIALOG	ISCONTROLSET	ISCONTROLSET.cbl:3160
MAIN	PRINTGUI	PRINTGUI.cbl:572
PRINT-INTERFACE	PRINTGUI	PRINTGUI.cbl:657

In addition to the “Quick Watch” dialog all other dialogs that contain the “Variable name” field, such as “Display variable”, “Modify variable”, “Display offset of variable”, “Display length of variable” and “Set monitor” have this new “Method/Program” field enabling developers to use this feature without the need to reload the previous source. This is useful when developers can specify names of the data items in previous programs directly.

The commands used in the Debugger command-line now have the option -c to specify the class name where the data item is declared, for example:

```
dis -c ISCONTROLSET current-page
let -c ISCONTROLSET current-page = 1
```

or, in case of CLASS-ID sources:

```
dis -c MyClass:>MyMethod myvar
let -c MyClass:>MyMethod myvar = abc
```

These commands will display and modify the values of the current-page variable in ISCONTROLSET program and the myvar variable in the MyMethod of MyClass.

Perform stack and Breakpoints

The Perform stack view and the output produced by the “infostack” command has been enhanced when using methods and entry-points. To better identify the current method or entry-point name in the Perform stack view, the syntax “of” has been added in the “Program” column. This is similar to the name of paragraphs shown in the Paragraph column when there are section names as well. For example:

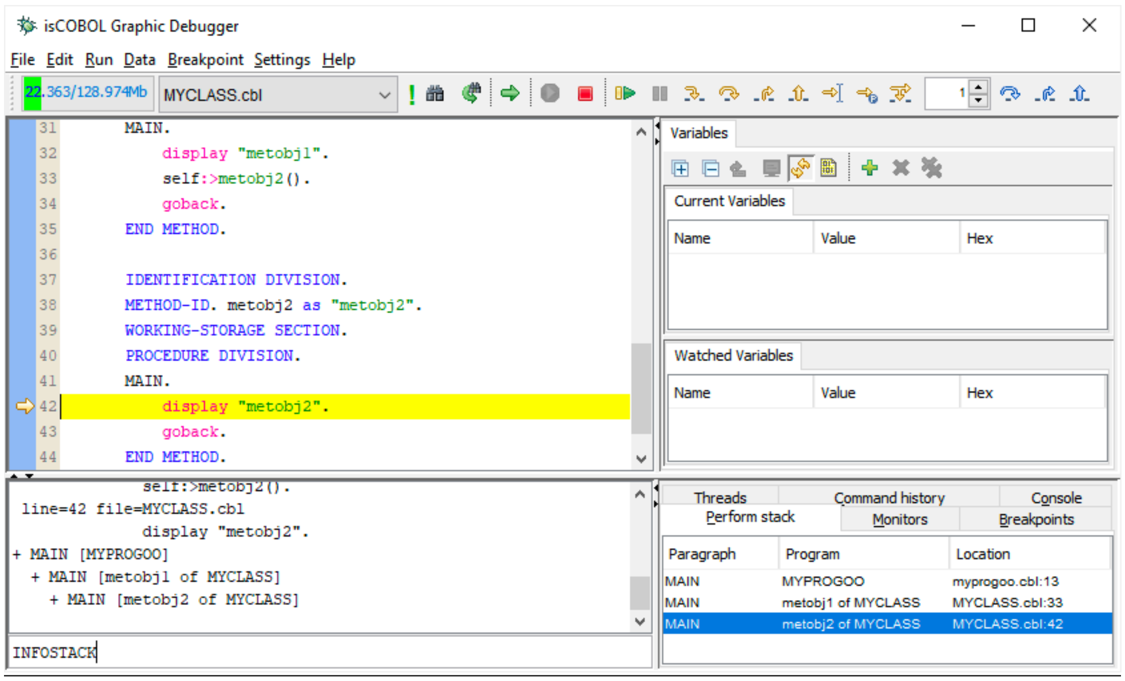
- “LOAD of SECT1” in the Paragraph means “LOAD” paragraph in “SECT1” Section

- “LOADLIB of MYLIB” in the Program means “LOADLIB” entry-point in “MYLIB” Program-id

- “setDate of MyClass” in the Program means “setDate” method in “MyClass” Class-id

As shown in Figure 3, *Enhancements to Breakpoints*, the output of the “infostack” command has also been improved with the same “of” syntax.

Figure 3. Enhancements to Breakpoints



```
       MAIN.
           display "metobj1".
           self:>metobj2().
           goback.
       END METHOD.

       IDENTIFICATION DIVISION.
       METHOD-ID. metobj2 as "metobj2".
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
       MAIN.
           display "metobj2".
           goback.
       END METHOD.
```

```
             self:>metobj2().
 line=42 file=MYCLASS.cbl
             display "metobj2".
+ MAIN [MYPROGOO]
  + MAIN [metobj1 of MYCLASS]
    + MAIN [metobj2 of MYCLASS]
```

```
INFOSTACK
```

Paragraph	Program	Location
MAIN	MYPROGOO	myprogoo.cbl:13
MAIN	metobj1 of MYCLASS	MYCLASS.cbl:33
MAIN	metobj2 of MYCLASS	MYCLASS.cbl:42

In addition, breakpoints set on “program”, “method” and “paragraph” level are maintained between different Debugger sessions also in case that the number of lines in the source is changed.

Graphical interface

isCOBOL 2024 R2 introduces a new control named CHIPS-BOX, a container that shows a list of chips in a box. Additional enhancements involve W$BITMAP and configurations to let developers customize and centralize window creation and control events.

CHIPS-BOX

The chips-box is typically used in web environments to let users pick a set of values from a predefined list, or to add new values not in the list. One of the most common uses of chips boxes is "tagging", where the user can pick existing tags or make up new ones.

In the 2024 R2 release isCOBOL implements this new control, using the syntax shown below:

```
SCREEN SECTION.
...
    03 chips chips-box chips-type 2 chips-radius radius-factor
       line 8 col 2 size 68 lines 10
       event CHIP-EVT.
...
PROCEDURE DIVISION.
...
    modify chips item-to-add item-text
                 bitmap image-handle(idx)
                 bitmap-number 1 bitmap-width 78-bmp-width
                 item-foreground-color 78-chip-color-1
                 item-background-color 78-chip-color-2
                 item-border-color -946895
                 item-rollover-background-color -14019325
                 item-rollover-foreground-color -678063
                 item-rollover-border-color -678063
                 hidden-data hidden-chip
                 giving item-id
```

The snippet declares a chip box element in the screen section, and in the procedure division the MODIFY statement adds a new chip with a leading bitmap.

The container can be customized using the following properties:

- CHIPS-TYPE can be set to 1 for chips that include just text and an optional leading bitmap, or 2 for chips that include text, an optional leading bitmap and a trailing x icon that the user can click to remove the chip.

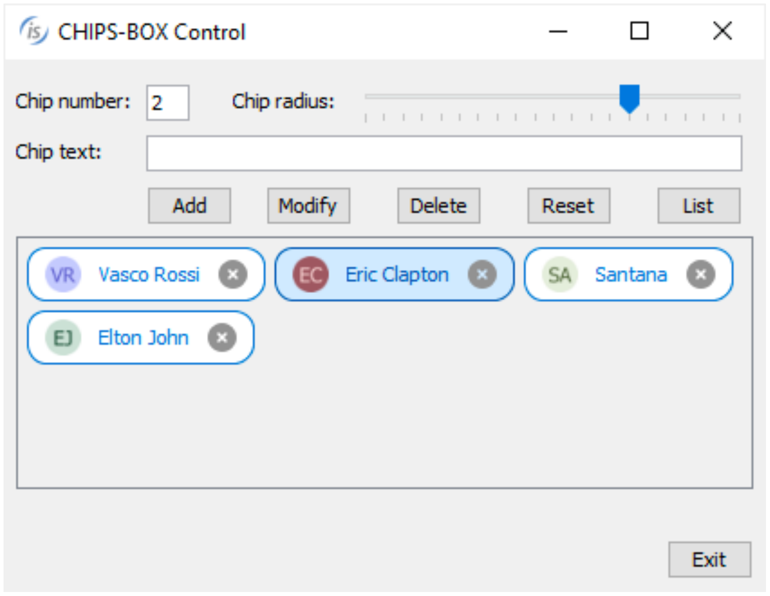
- CHIPS-RADIUS to set the percentage of the rounded borders, from 0 to have sharp borders to 100 to have circular borders.
- CHIPS-BORDER-WIDTH to set the width of the chips' borders.
- CHIPS-ROLLOVER-BORDER-WIDTH to set the width of the chips' borders when the mouse is over the chip.

Colors can be set on the chips themselves, or can be set as default on the container using the specific properties:

- ITEM-COLOR, ITEM-FOREGROUND-COLOR and ITEM-BACKGROUND-COLOR to set the color of the chip.
- ITEM-BORDER-COLOR to set the border color of the chip.
- ITEM-ROLLOVER-COLOR, ITEM-ROLLOVER-BACKGROUND-COLOR and ITEM-ROLLOVER-FOREGROUND-COLOR to set the color of the chip when the mouse is over the chip.
- ITEM-ROLLOVER-BORDER-COLOR to set the border color of the chip when the mouse is over the chip.

The result of running the program is shown in Figure 4, *New CHIPS-BOX control*, and the user is selecting the second chip, which is then painted using the rollover colors set.

Figure 4. New CHIPS-BOX control.



When the user clicks on the chip, the CMD-CLICKED event is fired, and when the X icon is clicked, the MSG-CLOSE event is fired. Developers can implement code to react to such events.

W$BITMAP

The new WBITMAP-TEXT-BOX opcode has been implemented in the W$BITMAP routine to generate images from text. The typical usage scenario is to display an avatar for a user account: if a user does not have an image selected for the account, an image is shown with the user's initials.

This feature is also used in the previous program for CHIPS where the leading bitmap is created dynamically by the program.

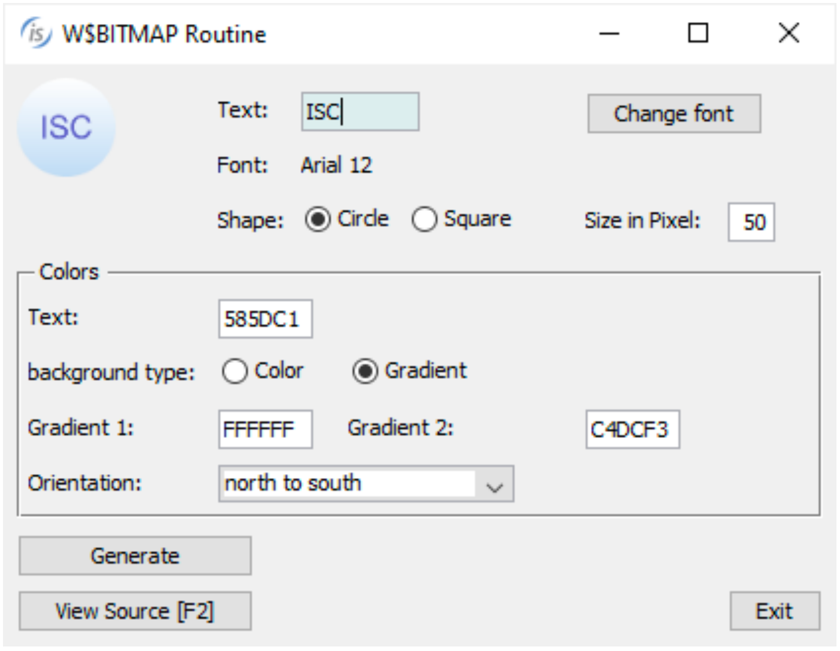
This code snippet:

```
initialize wbitmap-tb-data
set        wbitmap-tb-circle to true
move h-font                  to wbitmap-tb-font
move p-size                  to wbitmap-tb-width
move wrk-text-color          to wbitmap-tb-text-color
move wrk-back-color          to wbitmap-tb-bg-color
move wrk-grad-color          to wbitmap-tb-bg-color-2
move gradient-north-to-south to wbitmap-tb-grd-or
move "ISC" to p-text
call "w$bitmap" using wbitmap-text-box
                      p-text
                      wbitmap-tb-data
               giving h-image-icon
```

can be used to create an image with the text and color settings passed in the structure wbitmap-tb-data, and the resulting bitmap handle h-image-icon can be used in any control that supports a bitmap handle.

The results of the program in execution are shown in Figure 5, *W$BITMAP wbitmap-text-box op-code*, where the text “ISC” is represented in the circle image.

Figure 5. W$BITMAP wbitmap-text-box opcode.



Text: ISC
Font: Arial 12
Shape: [✓ Circle] Square
Size in Pixel: 50

Colors
Text: 585DC1
background type: Color [✓ Gradient]
Gradient 1: FFFFFF
Gradient 2: C4DCF3
Orientation: north to south

Other GUI enhancements

The entry-field proposal feature has been enhanced by adding a new PROPOSAL-FILTER-TYPE property that can be used to customize filtering of the proposal list of Entry-Field, and the possible values can be:

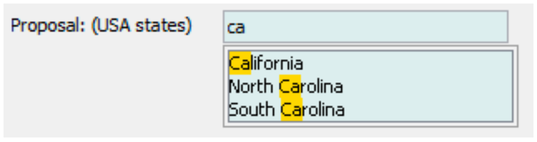
0: no filtering is performed, and the list is always shown entirely
1: filters with a case insensitive "contains" logic
2: filters with a case insensitive "starts with" logic

In addition, the text that matches is highlighted in the list. The following is a code snippet that shows usage the new property:

```
03 ef-state entry-field value w-state
   line 5, col 20, size 20 cells
   proposal-filter-type 1.
```

In Figure 6, *PROPOSAL-FILTER-TYPE on entry-field*, shows the results when running with filtering in progress.

Figure 6. PROPOSAL-FILTER-TYPE on entry-field.



The window supports a new WINDOW-STATE property that can be used in the INQUIRE statement to detect the state of the window, allowing the code to check if the window is minimized or maximized, as shown in this snippet of code:

```
inquire h-win window-state wstate
evaluate wstate
  when win-normal         ...
  when win-iconified      ...
  when win-maximized-both ...
end-evaluate
```

The event-data-1 and event-data-2 data items returned in the MSG-MOUSE-ENTER event fired for controls like grids, list-box and tree-view when the NOTIFY-MOUSE style is set now contain more detailed information. For example, in a grid the cell coordinates are included, and in a tree-view the mouse coordinates are included.

New configurations have been implemented to customize GUI control behavior:

- iscobol.gui.window.hook assigns a class to customize the DISPLAY WINDOW behavior. The class needs to implement the isCOBOL interface "com.iscobol.rts.WindowCreateHandler". Developers can inquire and modify attributes before or after window creation.

  The following is a code snipped of the class-id source:

```
IDENTIFICATION DIVISION.
CLASS-ID. WCWINHANDLER AS "WCWINHANDLER" IMPLEMENTS WINCREATEHANDLER.
...
    CLASS WINATTRIBUTEHOOK AS "com.iscobol.gui.server.WindowAttributeHook"
    CLASS WINCREATEOVEXC   AS "com.iscobol.rts.WindowCreateOverflowException"
    CLASS WINCREATEHANDLER AS "com.iscobol.rts.WindowCreateHandler"
...
METHOD-ID. IS-WINDOWCREATE AS "beforeWindowCreate" override.
...
LINKAGE SECTION.
77  MyWinAttribute OBJECT REFERENCE WINATTRIBUTEHOOK.
procedure division using MyWinAttribute raising WINCREATEHANDLER.
if env-code = runenv-web-client
   set win-type to MyWinAttribute:>getType
   if  win-type = "INDEPENDENT"
       MyWinAttribute:>setBackground(-16054009)
       MyWinAttribute:>setUndecorate(true)
   end-if
end-if
...
METHOD-ID. IS-AFTERWINDOWCREATE AS "afterWindowCreate" override.
...
LINKAGE SECTION.
77  myWinhandler OBJECT REFERENCE WINCREATEHANDLER.
procedure division using myWinhandler raising WINCREATEHANDLER.
...
```

  When running using the configuration:

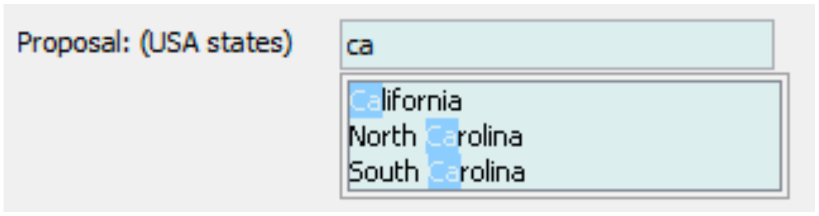
```
iscobol.gui.window.hook=WCWINHANDLER
```

  the class is invoked for every window created. If the window type is INDEPENDENT and the runtime environment is WebClient then the window is colored differently and set to UNDECORATE. This feature allows extensive control to windows. For example, a hook program can be used to customize the appearance of an application depending on the environment the program is run on.

- iscobol.gui.matching_text_color specifies a matching color in entry-field, list-box and grid controls. When running using the configuration:

  ```
  iscobol.gui.matching_text_color=-10079487,-14675438
  ```

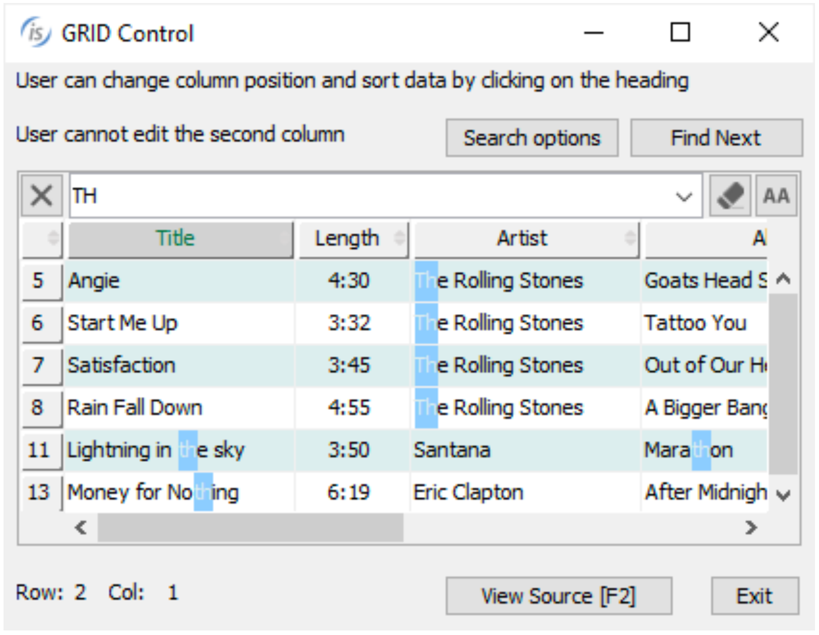
  the selected text in the entry-field proposal and in the search result of list-box and grid controls is painted using the provided combination of RGB colors as background/foreground text color.



Proposal: (USA states)

ca

California
North Carolina
South Carolina



GRID Control

User can change column position and sort data by clicking on the heading

User cannot edit the second column

TH

	Title	Length	Artist	A
5	Angie	4:30	The Rolling Stones	Goats Head S
6	Start Me Up	3:32	The Rolling Stones	Tattoo You
7	Satisfaction	3:45	The Rolling Stones	Out of Our H
8	Rain Fall Down	4:55	The Rolling Stones	A Bigger Ban
11	Lightning in the sky	3:50	Santana	Marathon
13	Money for Nothing	6:19	Eric Clapton	After Midnigh

Row: 2 Col: 1

- iscobol.hot_event.<program-name>=<event-type(s)> provides a custom handler for specific GUI events. The program-name is a program-id that receives an event-status structure as parameter in linkage section.

  For example, a program like this:

```
program-id. EFXICON.
working-storage section.
copy "iscontrols.def".
77 w-class    pic 99.
linkage section.
copy "iscrt.def".
procedure division using event-status.
MAIN.
    inquire event-control-handle class in w-class
    if w-class = ctl-entry-field
       if event-data-1 = 2   | click on the trailing bitmap (Xmark)
          modify event-control-handle value ""
       end-if
    end-if
    goback 0.
```

  with the configuration set using:

```
iscobol.hot_event.EFXICON=16400
```

  will enable a developer to centrally manage the events msg-bitmap-clicked for the entire application.

  The program EFXICON will be executed and it will manage the event with specific code just for clicking on trailing bitmaps in entry-field controls.

  This solution can be easily adopted when using code injection. For example to activate the X icon as a trailing bitmap in all entry-fields, just compile with this configuration:

```
iscobol.compiler.gui.entry_field.defaults=bitmap-handle h-tools \
                                          bitmap-trailing-number 78-nb-xmark \
                                          bitmap-width  78-tb-bmp-width \
                                          event evt-ef-empty
```

  This approach of centralized handling is similar to what can be achieved using the configuration iscobol.hot_key.PROGNAME=n to manage the function keys or any exception value.

Database Bridge enhancements

isCOBOL Database Bridge is the tool that enables COBOL programs to interact with a RDBMS without changing COBOL source code or learning ESQL. It has been enhanced to optimize interactions with Microsoft SQL Server.

The compiler property iscobol.compiler.easydb.light_cursor can now be used along with iscobol.compiler.easydb.sqlserver as follows:

```
iscobol.compiler.easydb=true
iscobol.compiler.easydb.sqlserver=true
iscobol.compiler.easydb.light_cursors=2
```

EDBI routines generated with this configuration retrieve *n* records at a time instead of retrieving the whole resultset at once. The size of the block of received records is controlled by the runtime configuration property iscobol.easydb.sqlserver.row_limit, whose default value is 100. It can be changed dynamically by SET ENVIRONMENT statement just before executing START file I/O statement.

Reading records with this logic reduces the workload on the database and improves performance when there are several runtime sessions working on the database.

The difference between light_cursor set to 1 instead of 2 is to use the pagination logic only when using UNIQUE indexes.

For developers that prefer to generate EDBI routines with the legacy two-steps approach by processing the EFD dictionaries with the edbiis command, these new edbiis options are now available:

-dslu (equivalent of iscobol.compiler.easydb.light_cursors=1)

-dsld (equivalent of iscobol.compiler.easydb.light_cursors=2)

For example, the command to be used for the legacy approach is:

```
edbiis -dsld FileName.xml
```

At runtime the application can improve performance by lowering the number of returned rows in the result set, example by setting:

```
iscobol.easydb.sqlserver.row_limit=50
```

Compatibility enhancements

isCOBOL 2024 R2 release improves the compatibility with other COBOL dialects such as ACUCOBOL-GT© and RM/COBOL©. Additionally, the ESQL syntax has been enhanced to improve the compatibility with DB2 Preprocessor.

COBOL compatibility

New library routines have been added:

- The P$SETBOXSHADE routine, implemented for RM compatibility, sets the color and density to be used by P$DRAWBOX. This is useful for graphical print output for customers that use the CALL P$* series of routines. The following code snippet shows the usage of the new routine:

```
CALL "P$SETBOXSHADE" USING "Black", 10
CALL "P$DRAWBOX" USING 2.5, 3.0, "Absolute", "Inches", .25, .25,
                       "Inches", "No".
CALL "P$SETBOXSHADE" USING "Red", 20
CALL "P$DRAWBOX" USING 3.5, 3.0, "Absolute", "Inches", .25, .25,
                       "Inches", "Yes".
```

  to create 2 boxes; the first is black and not filled, the second is red with low intensity and filled inside.

- The C$REGEXP routine, implemented for ACU compatibility, is used to search strings using regular expressions. Though using isCOBOL's OOP syntax to use java.util.regex.* is more powerful, the new routine makes it easier to port code as-is when migrating from ACUCOBOL-GT Extend code. The routine has different op-codes to compile a regexp, check for matches in a string, or for every group when performing group checks. If an error occurs detailed information is available using the CREGEXP-LAST-ERROR opcode and the memory allocated for handles can be released using the CREGEXP-RELEASE-MATCH and CREGEXP-RELEASE op-codes.

This is a code snippet of the new routine:

```
...
77 ret-regexp  pic s99.
77 w-string    pic x(50).
77 reg-expr    pic x(50).
77 h-regexp    handle.
77 h-result    handle.
...
CALL "C$REGEXP" USING CREGEXP-GET-LEVEL GIVING ret-regexp
move "This is a big    house with garden" to w-string
string "(big)\s+(house)" x"00" delimited by size into reg-expr
CALL "C$REGEXP" USING CREGEXP-COMPILE, reg-expr
                GIVING h-regexp
if h-regexp not = null
   move 0 to w-length, match-start, match-end
   CALL "C$REGEXP" USING CREGEXP-MATCH, h-regexp, w-string,
                         w-length, match-start, match-end
                   GIVING h-result
   if h-result = 0
      CALL "C$REGEXP" USING CREGEXP-LAST-ERROR GIVING ret-regexp
   else
      compute w-bytes = match-end - match-start
      display "found:" w-string(match-start:w-bytes)
   end-if
   CALL "C$REGEXP" USING CREGEXP-NUMGROUPS h-result
                   GIVING ret-regexp
   move ret-regexp to num-groups
   perform varying w-group from 1 by 1 until w-group > num-groups
      CALL "C$REGEXP" USING CREGEXP-GETMATCH, h-result,
                            w-group, idx-start, idx-end
                      GIVING ret-regexp
      ...
   end-perform
end-if
CALL "C$REGEXP" USING CREGEXP-RELEASE-MATCH h-result
CALL "C$REGEXP" USING CREGEXP-RELEASE h-regexp
...
```

that shows how to use the different op-codes to perform searches.

ESQL compatibility

IsCOBOL 2024 R2 contains many enhancements in ESQL. It's now possible to use group data items as parameter of an IN clause. In addition, the compatibility with the IBM DB2 syntax when compiling using the -csdb2 compiler option has been improved with the support of VALUES INTO and DECLARE VARIABLE statements.

When executing the sql statement

```
EXEC SQL
   SELECT field-list FROM table-name
          WHERE field IN (value1, value2, ..., valueN)
END-EXEC
```

in versions up to 2024 R1, you had to use separate host variables for value1, value2 and valueN, e.g.

```
01 wk-value1 pic x(n).
01 wk-value2 pic x(n).
01 wk-value3 pic x(n).
```

Starting from the 2024 R2 release a single group host variable can be used instead:

```
01 wk-values.
   03 wk-value1 pic x(n).
   03 wk-value2 pic x(n).
   03 wk-value3 pic x(n).
```

The sub items will be used to set value1 to valueN.

The VALUES INTO statement produces a result table consisting of at most one row and assigns the values of columns in that row to host variables. For example:

```
EXEC SQL
   VALUES(CURRENT PATH) INTO :hvl
END-EXEC.
```

This statement is supported when using the -csdb2 compiler option and is internally translated to:

```
EXEC SQL
   SELECT CURRENT PATH INTO :hvl FROM SYSIBM.SYSDUMMY1
END-EXEC.
```

The DECLARE VARIABLE statement defines a CCSID for a host variable and the subtype of the variable. When it appears in an application program, the DECLARE VARIABLE statement causes the compiler to tag a host variable with a specific CCSID. When the host variable appears in an SQL statement, the compiler places this CCSID in the structures that it generates for the SQL statement.

Some examples:

```
EXEC SQL
   DECLARE :W1AX-FG-NR-EBC VARIABLE CCSID EBCDIC
END-EXEC.
```

specifies that the default EBCDIC CCSID for the type of the variable at the server should be used.

```
EXEC SQL
   DECLARE :W1AX-FG-NR-ASC VARIABLE CCSID ASCII
END-EXEC.
```

specifies that the default ASCII CCSID for the type of the variable at the server should be used.

```
EXEC SQL
   DECLARE :W1AX-FG-NR-BIT VARIABLE FOR BIT DATA
END-EXEC
```

specifies that the values of the host-variable are not associated with a coded character set and therefore are never converted.

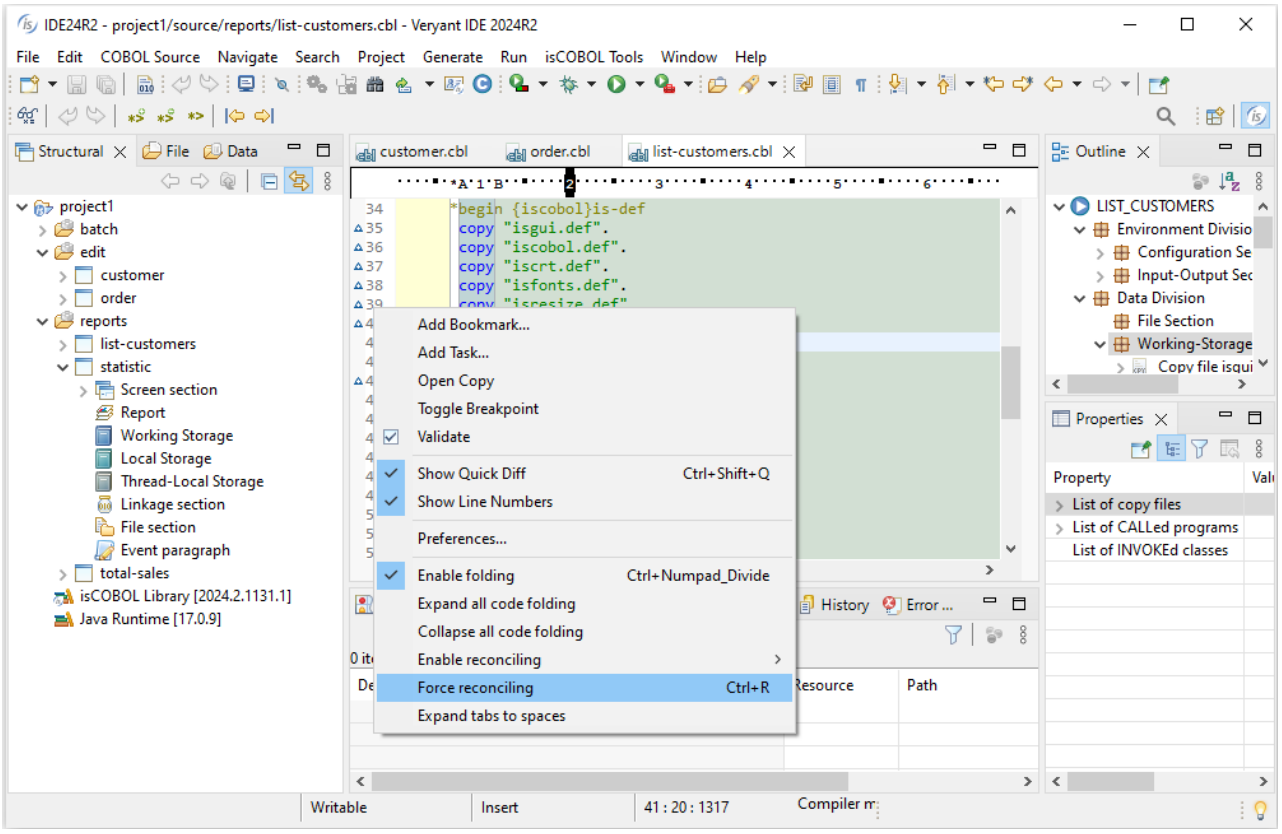
isCOBOL IDE

isCOBOL IDE 2024 R2 supports subfolders in the “Structural view” used for Screen programs or WOW programs and in the “Data view” used for FD/SL files managed in the painters. This enables developers to more effectively organize a project, keeping the programs of different categories in different folders.

In addition, the Reconciling feature can now be executed on demand with a menu item or by using the corresponding hot key combination when the developer needs it. This is useful when the Reconciling feature is not always activated, allowing it to be executed as needed.

In Figure 7, *IDE subfolders and Force reconciling*, the new features are shown.

Figure 7. IDE subfolders and Force reconciling.



```
*begin {iscobol}is-def
 copy "isgui.def".
 copy "iscobol.def".
 copy "iscrt.def".
 copy "isfonts.def".
 copy "isresize.def"
```

Additional improvements

The isCOBOL 2024 R2 release contains additional improvements on configuration properties, EIS Servlets, COBOL WOW optimization and EsqlRuntime class.

New configurations

The Profiler utility has been enhanced with the following new configurations:

- `iscobol.profiler.enable=false` to start the program with the profiler disabled. This is useful when running applications with a user interface and it's important to avoid profiling during user interaction. When a program that needs to be profiled is called, the called program can perform a CALL to the C$PROFILER library to enable the profiler, providing more accurate results.
- `iscobol.profiler.elapse_time=n to` sets a timeout in seconds for a profiler flush. This is useful when running batch programs that take a long time but for the developer's analysis it's enough to collect the first n seconds or minutes of the execution. With this configuration set, after the elapsed time the profiler flushes the results, and the program keeps running until completion.

The CALL PROGRAM has been enhanced with a new configuration:

- `iscobol.call_program.set_switches=true` to set switches in CALL PROGRAM statement with syntax /A/B. When this configuration is set a statement like this:
  ```
  CALL PROGRAM "MYPROG/A/D".
  ```
  executes the MYPROG program and activates its switch "A" and switch "D".

  Without this option, performing the CALL above would try to invoke program D inside a folder MYPROG/A.

EIS Servlets

EIS Servlets manage sessions using what's called "jsessionid" which acts as an identifier for the session. It is usually stored as a cookie in the browser, but in some scenarios that may not be the desired way. For example, multiple tabs on the same page share cookies so all tabs share the same session on the server.

To provide independent tabs a different session tracking solution is needed.

By modifying the web.xml file associated with the web application, the tracking mode can be overridden, as shown by the code snippet below.

```
<session-config>
     <tracking-mode>URL</tracking-mode>
</session-config>
```

With this configuration, the session tracking mode switches from cookies to URL. isCOBOL EIS Servlet checks the invoked URL for the presence of a jsessionid query parameter, and executes an URL redirection if none is found, appending the string ";jsessionid=<sessionId>" to the original URL where <sessionId> is the id of the session. The setting written in the web.xml file causes the creation of a new session of each call to our servlet that lacks the jsessionid parameter but will switch to the provided session when a matching sessionid is located.

Each browser tab can then specify a different jsessionid parameter allowing different sessions to be used.

Be mindful when using such a feature, since the session id will be visible in the URLs and can be replicated with copy and paste and stored in proxy server logs, web server logs and browser history. This could allow an attacker to grab a valid session ID and get access to your users' sessions.

COBOL WOW optimization

Applications migrating from RM/COBOL WOW can now be optimized by calling the new WOWSTARTBUFFERING and WOWSTOPBUFFERING routines to enable the buffering system during WOW calls. This is especially useful when running in ThinClient or WebClient environments, as it reduces the TCP communication overhead to improve responsiveness. After the WOWSTARTBUFFERING routine is called all following CALLs to WOW routines are buffered by the isCOBOL Server. When the WOWSTOPBUFFERING routine is called, the updates are sent to the client, causing the user interface to update and repaint as needed and providing a smoother user experience. This feature is similar to the corresponding MASS-UPDATE feature used for Screen programs.

A code snippet like this:

```
CALL WOWSTARTBUFFERING USING WIN-RETURN
CALL WOWSETPROP USING WIN-RETURN CT1 "VISIBLE" 0
PERFORM UNTIL EXIT
   ...
   CALL AXSETINDEXPROP USING WIN-RETURN CT1 "CellText" WCOL0 WLIN1 0
   CALL AXSETINDEXPROP USING WIN-RETURN CT1 "CellText" WCOL1 WLIN1 1
   CALL AXSETINDEXPROP USING WIN-RETURN CT1 "CellText" WCOL2 WLIN1 2
   ...
END-PERFORM
CALL WOWSETPROP USING WIN-RETURN CT1 "VISIBLE" 1
CALL WOWSTOPBUFFERING USING WIN-RETURN
```

loads an entire ctGrid with minimal TCP overhead.

EsqlRuntime class

The com.iscobol.rts.EsqlRuntime class contains a new method named getResultSet to retrieve the ResultSet object of a Cursor. This object can then be passed to methods in Java classes that require such objects. The method signature of the implemented method is:

```
public static ResultSet getResultSet(String cursorName)
```

and this is a code snippet on how to use it after the cursor is opened:

```
repository.
    class ESQLRuntime as "com.iscobol.rts.EsqlRuntime"
    class ResultSet   as "java.sql.ResultSet"
...
77  obj-rs   object reference ResultSet.
...
    EXEC SQL
         OPEN CUR
    END-EXEC.
    set obj-rs to ESQLRuntime:>getResultSet("CUR")
...
```