



# isCOBOL™ Evolve

## isCOBOL Evolve 2015 Release 1 Overview

Copyright © 2015 Veryant LLC.

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution and recompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Veryant and its licensors, if any.

Veryant and isCOBOL are trademarks or registered trademarks of Veryant LLC in the U.S. and other countries. All other marks are property of their respective owners.

## **isCOBOL Evolve 2015 Release 1 Overview**

### **Introduction**

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2015 R1.

isCOBOL Evolve provides a complete environment for the development, deployment, maintenance, and modernization of COBOL applications. isCOBOL 2015 R1 includes several enhancements to isCOBOL IDE, to the utilities and several optimizations in the isCOBOL Server product; isCOBOL 2015 R1 also includes many Debugger improvements, and other new features like isCOBOL profiling tool and more.

Details on these enhancements and updates are included below.

### **isCOBOL IDE Enhancements**

In isCOBOL Evolve 2015 R1, the isCOBOL IDE has been created with two different editions:

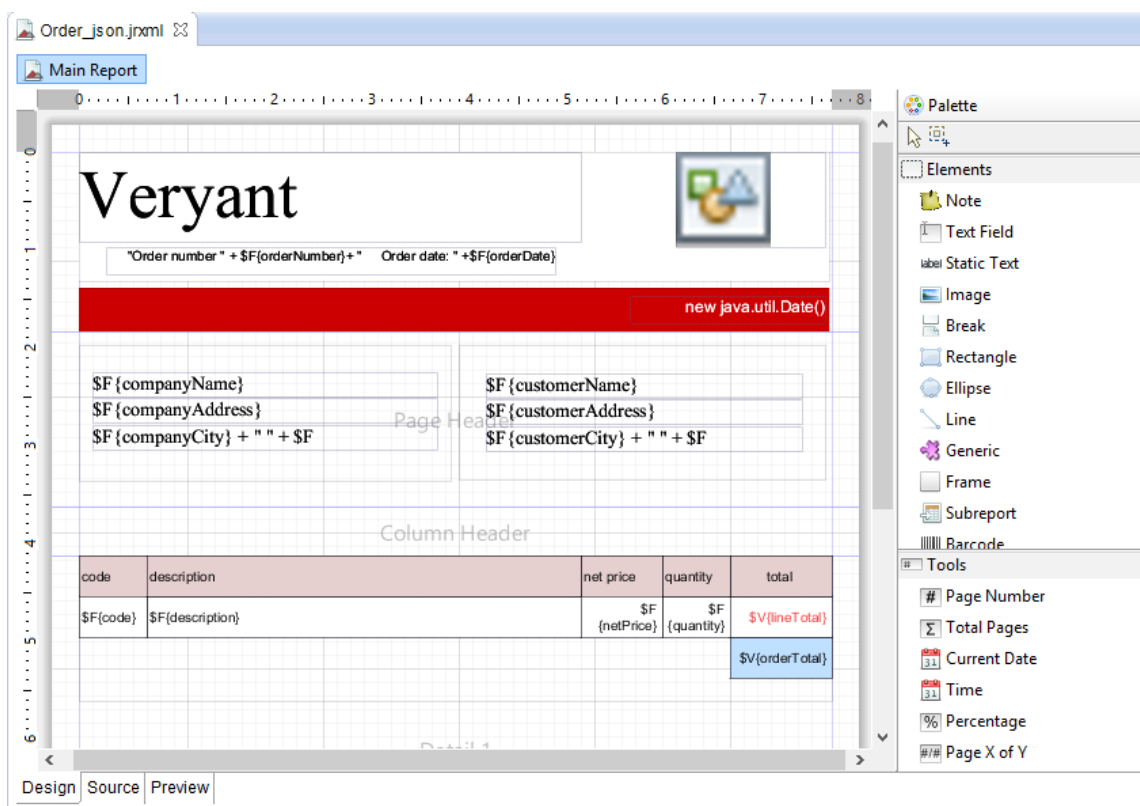
- Standard setup, that is based on Eclipse Kepler SR2 for Java EE with the inclusions of Jasperstudio plugins and others useful plugins
- Lite setup, that is based on Eclipse Kepler SR2 with essential plugins included.

Both editions contain all isCOBOL IDE plugins to manage isCOBOL projects, programs, screens, data files and reports.

## Jasper Report

The addition of Jasper report studio to the isCOBOL IDE, as shown in Figure 1, *Jasper Studio Designer*, allows isCOBOL users to create sophisticated layouts containing charts, images, sub-reports, crosstabs and much more. Moreover it enables you to access your data through JDBC, TableModels, JavaBeans, XML, Hibernate, CSV, and custom sources. Then to publish your reports as PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX, or OpenOffice

**Figure 1.** Jasper Studio Designer



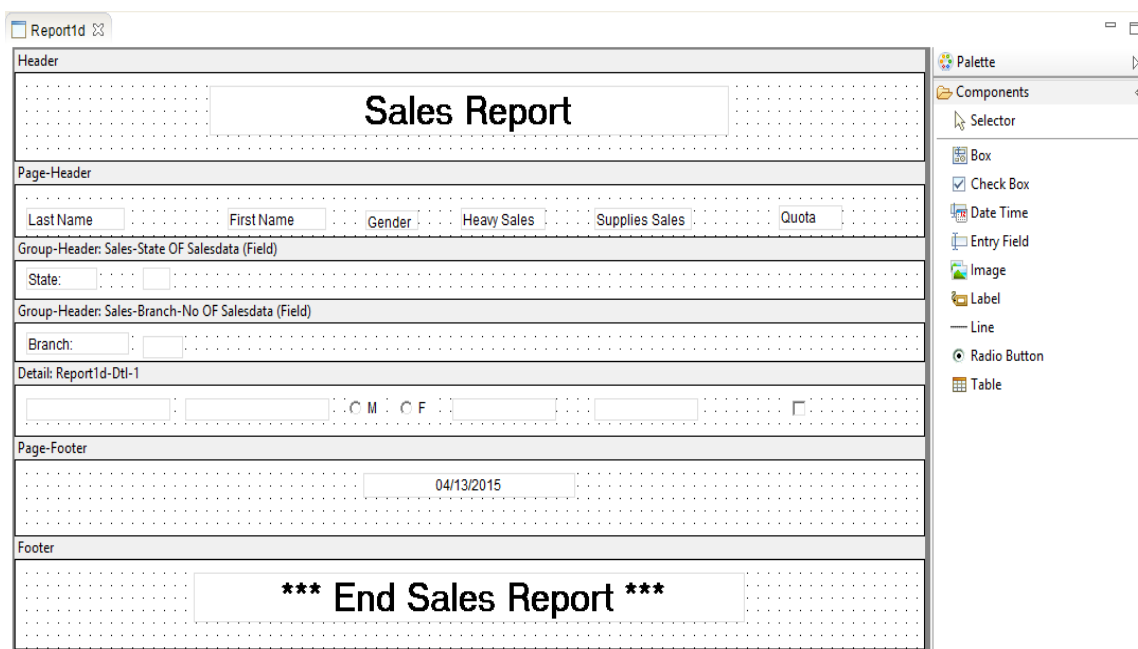
Using Jasper Report Studio to design report's applications, is the first step in taking advantage of the flexible BI architecture provided by JasperSoft now TIBCO.

## Report Designer

isCOBOL IDE 2015 R1 introduces a new feature to be able to generate brand new reports. In addition, you are now able to import AcuBench reports. Once AcuBench report is imported, isCOBOL IDE users can continue maintaining the reports using a WYSIWYG graphical design window where you can drag and drop report components from a graphical palette.

As shown in Figure 2, *isCOBOL IDE Report Designer*, once components are dragged into the report, you can use the Property window to configure the appearance and behavior of the element and the Event Editor to tie the code to the element.

**Figure 2.** isCOBOL IDE Report Designer

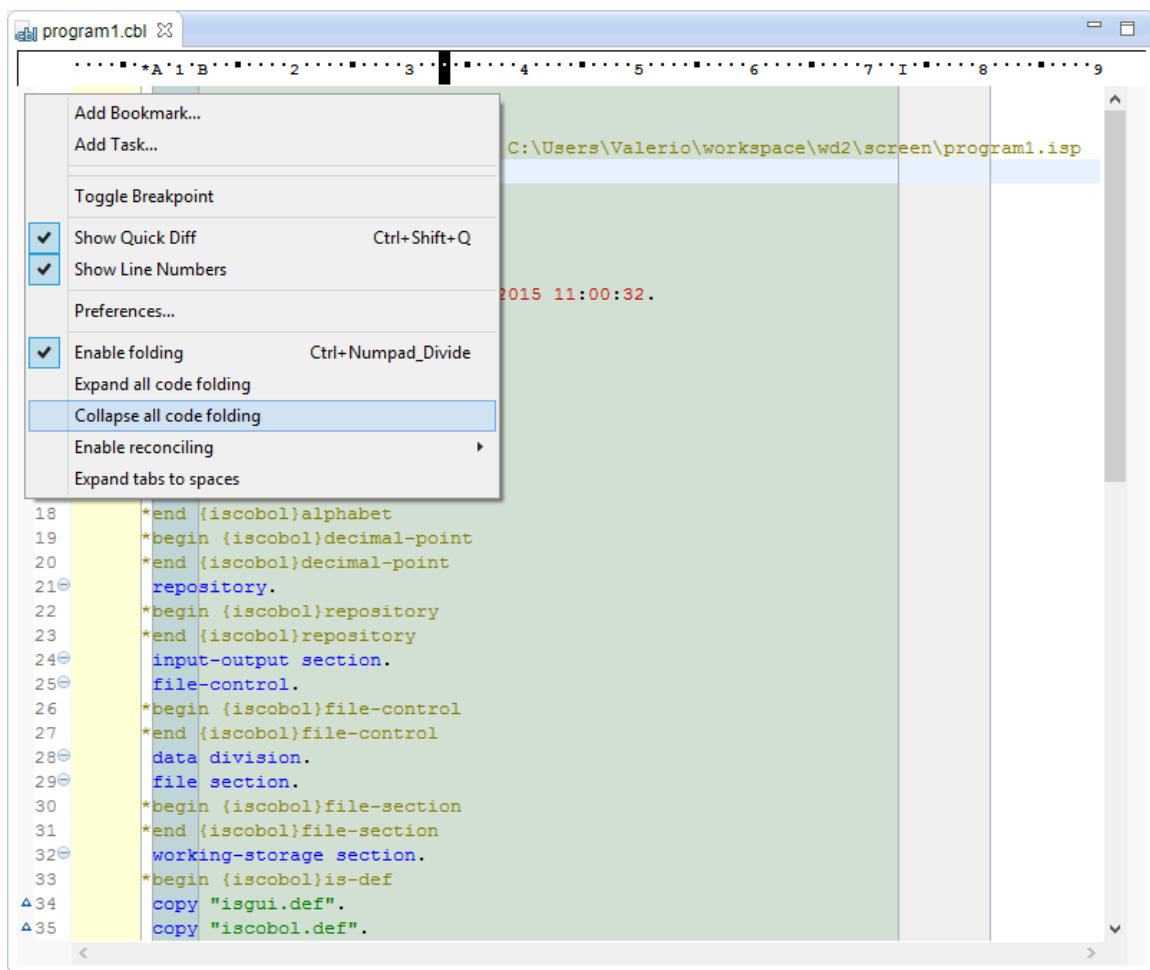


While the AcuBench reports are executed as HTML files to be printed via IE Active/X using the acubenchprint.dll wrapper, the isCOBOL support for AcuBench reports, includes an HTML rendering tool that is able to print and preview generated HTML without any IE Active/X dependences. However, is also possible to continue using acubenchprint.dll if needed.

### Enhanced code folding

Figure 3, *Collapse/Expand code folding*, shows two new functions 'Expand all code folding' and 'Collapse all code folding', to expand or collapse all sections of a COBOL source code. This allows the user to manage large amounts of code while viewing only subsections of the code that are specifically relevant at any given time.

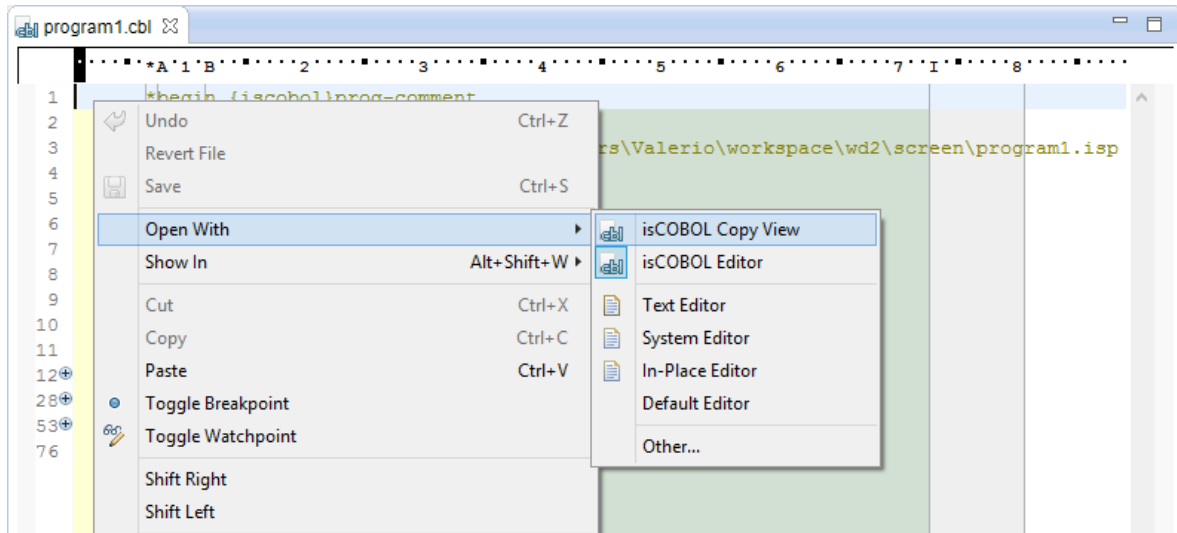
**Figure 3.** Collapse/Expand code folding



### IsCOBOL Copy View

It is useful to have a COBOL source with the inclusion of all copy files to make a full text search that includes also all used copy files. As depicted in Figure 4, *isCOBOL Copy View*, is a new entry on existent "Open With" menu.

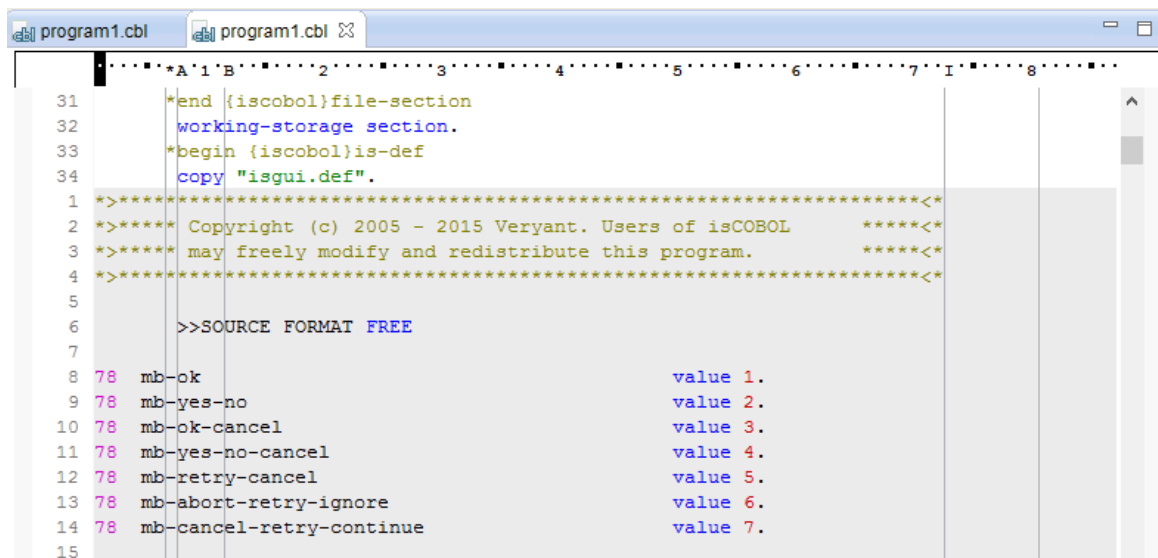
**Figure 4.** isCOBOL Copy View





When a user decides to open a COBOL source with 'isCOBOL Copy View' it creates a new view of the COBOL source that will be opened, where all copy files are included using a different background color as depicted in Figure 5, *new isCOBOL View including all copy files*.

**Figure 5.** New isCOBOL View including all copy files



The screenshot shows a window titled 'program1.cbl' with a COBOL source code editor. The code is displayed with a light gray background. The code includes a file-section, a working-storage section, and a copy statement for 'isgui.def'. The copy file content is shown with a yellow background. The code is as follows:

```
31      *end {iscobol}file-section
32      working-storage section.
33      *begin {iscobol}is-def
34      copy "isgui.def".
1  >>*****<<
2  >>***** Copyright (c) 2005 - 2015 Veryant. Users of isCOBOL *****<<
3  >>***** may freely modify and redistribute this program. *****<<
4  >>*****<<
5
6      >>SOURCE FORMAT FREE
7
8  78  mb-ok                               value 1.
9  78  mb-yes-no                           value 2.
10 78  mb-ok-cancel                         value 3.
11 78  mb-yes-no-cancel                     value 4.
12 78  mb-retry-cancel                      value 5.
13 78  mb-abort-retry-ignore                value 6.
14 78  mb-cancel-retry-continue             value 7.
15
```

### Ability to run the import processes from the command line

Added the ability to run the PSF and DLT import process outside the isCOBOL IDE.

#### Syntax for PSF:

```
isIDE.exe -data <workspaceLocation> -nosplash -application  
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importApplication project  
<projectname>  
    {file <file-import-program>}  
    {folder <folder-path>}  
    {logfile <logFile>}  
    {cellWidthFact <cell-width>}  
    {cellHeightFact <cell-height>}
```

where project and file or folder are required

If the <logFile> is given it is used for the log otherwise the default log file is used  
(iscobol\_import<date-time>.log)

Example command:

```
isIDE.exe -data C:\WorkspaceDST -nosplash -application  
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importApplication project ProjectDST  
folder C:\ISCOBOLtests\psffolder
```

#### Syntax for DLT:

```
isIDE.exe -data <workspaceLocation> -nosplash -application  
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importDltApplication project  
<projectname>  
    {file <file-import-program>}  
    {folder <folder-path>}  
    {logfile <logFile>}
```

where project and file or folder are required

If the <logFile> is given it is used for the log otherwise the default log file is used  
(iscobol\_importDLT<date-time>.log)

Example command:

```
isIDE.exe -data C:\MyWorkspace -nosplash -application  
com.iscobol.plugins.screenpainter.IscobolScreenPainter.importDltApplication project Project  
folder C:\dltfolder
```

## isCOBOL Compiler Enhancements

The isCOBOL Evolve 2015 R1 release includes many changes on the isCOBOL Compiler that improve its power and flexibility simplifying some areas of migration to isCOBOL from other COBOLs.

### Index search for similar words

The START statement was improved to allow the search of similar keys. For example: If a user searches an index file for the word "Verynt" and there is a word "Veryant" in the index the search engine or the program may ask the user the question "Do you mean Veryant ? " as it happens with most search engines on the web.

The implementation uses a Damerau-Levenshtein distance algorithm to calculate the "distance" between two strings and determine how many steps it takes to transform one string into another. APPROX a new clause is supported in the LIKE operator to allow searches with approximate string matching.

Example:

```
77 w-company-name pic x any length.
77 w-approx-lev   pic 9.
...
move "Verynt" to w-company-name
move w-company-name to company-name
move 1 to w-approx-lev
start file-company key > company-name while
    like trimmed APPROX w-approx-lev
    case-insensitive company-name
perform until fstatus not = "00"
    read file-company next not at end
        if company-name
            like trimmed approx w-approx-lev
            case-insensitive w-company-name
            display "found the company: " company-name
        end-if
    end-read
end-perform
```

The above example will start the file containing the company names with the APPROX syntax, and it will check with the IF statement if the company-name match with the given string. So "Veryant" will be a good result because it is approximate to "Verynt".

### Object Oriented syntax improvements

The INVOKE and OBJECT REFERENCE syntax are now enhanced to support the dynamic method of name invocation. The implementation follows ANSI2002 rules on "Universal Object" argument. In practice, declaring an OBJECT REFERENCE without the ClassName, makes the object become a Universal Object. The method name invoked can then be set in a variable to take advantage of the dynamic method name.

Code example:

```
repository.  
  class cstring as "java.lang.String"  
  class cinteger as "java.lang.Integer"  
  .  
working-storage section.  
01 ostr object reference cstring.  
01 universalObject object reference.  
01 wsmethodname pic x(20).  
procedure division.  
  set ostr = "hello world"  
  move "substring" to wsmethodname  
  invoke ostr wsmethodname using 2 8 returning universalObject  
  display universalObject:>toString  
  set universalObject to null  
  set universalObject to cinteger:>new(3)  
  display universalObject:>toString  
  .
```

### New compiler options

The isCOBOL 2015 R1 compiler introduces new options

- -ssnl to convert subroutine names to lower case
- -ssnu to convert subroutine names to upper case
- -cnlz to have leading zeros shown in character numeric display
- -cfp36 allows the intermediate results to always be calculated to 36 digits
- -wdbz shows warnings for possible divide by zero without ON SIZE ERROR

-ssnl and -ssnu, could be useful during MicroFocus migrations when C routines are called. Thereby, it allows those options at the compile time to force upper/lower case name of C routines to be called.

-cnlz helps all character base applications that need to have leading zeros shown when a numeric field is displayed on the screen. The isCOBOL default behavior is to remove leading zeroes during DISPLAY statement. Some COBOL implementers have different default behavior so these options simplify the migration to isCOBOL.

-c36 forces isCOBOL runtime to make intermediate calculations using 36 digits instead of 20 digits. COBOL programs that need calculation and better precision can take advantage of this option.

-wdbz produces a warning to easily identify cases where COMPUTE or DIVIDE statements are used without ON SIZE ERROR clause. Having COMPUTED or DIVIDE statement without ON SIZE ERROR could produce unexpected results or runtime errors in case of divide by zero.

### Enhanced syntax

ACCEPT FROM WINDOW allows that specifying OF the H-THREAD to receive the window handle from specific thread. This allows you to easily retrieve the window handle of a separated thread (for example a called program in thread mode) from a different running thread (for example the main menu of the application) to set the input window as required.

Example:

```
accept hwin-pgm from window of thread-handle
set input window to hwin-pgm
```

### Enhanced External File Descriptors (.xml or .iss)

Complex conditions are now fully supported with AND, OR syntax.

Examples:

```
*(( efd when field-type = 1 or
      field-type = 2 and
      field-zone = 1 tablename=custzone1 ))
..
*(( efd when field-type = other and
      field-cust >= 100 tablename=othercust ))
```

New property to define default Julian base date

A new property named `iscobol.compiler.iss_julian_base` was provided to setup the default base date for all Julian date defined.

For example if you have defined some Julian date on FD like:

```
$EFD DATE=JJJJJ
      03 arc-d pic 9(5).
```

Setting in isCOBOL properties file a property like:

```
iscobol.compiler.iss_julian_base=19000101
```

This means that all Julian dates will use 1 January 1900 as Julian base date. If the property is not set, no date is registered in the iss dictionary and so CTreeSQL will use the 1 March 1700.

### Enhanced compatibility with other COBOLs

In addition to the above compiler options, starting from isCOBOL 2015 R1, ESQL built-in pre-compiler, now supports square brackets and dots in stored procedure names.

A new library routine named C\$XML was added to simplify migration from ACUCOBOL-GT. Starting from the beginning, isCOBOL was equipped with a powerful and native way to support XML streams.

## Debugger Improvements

### Expand and Collapse copy files

Now the isCOBOL Debugger allows you to expand copy files to see all the COBOL code. As depicted in Figure 6, *New Expand and Collapse*, a “plus” symbol allows you to expand or collapse related copy files.

**Figure 6.** New Expand and Collapse

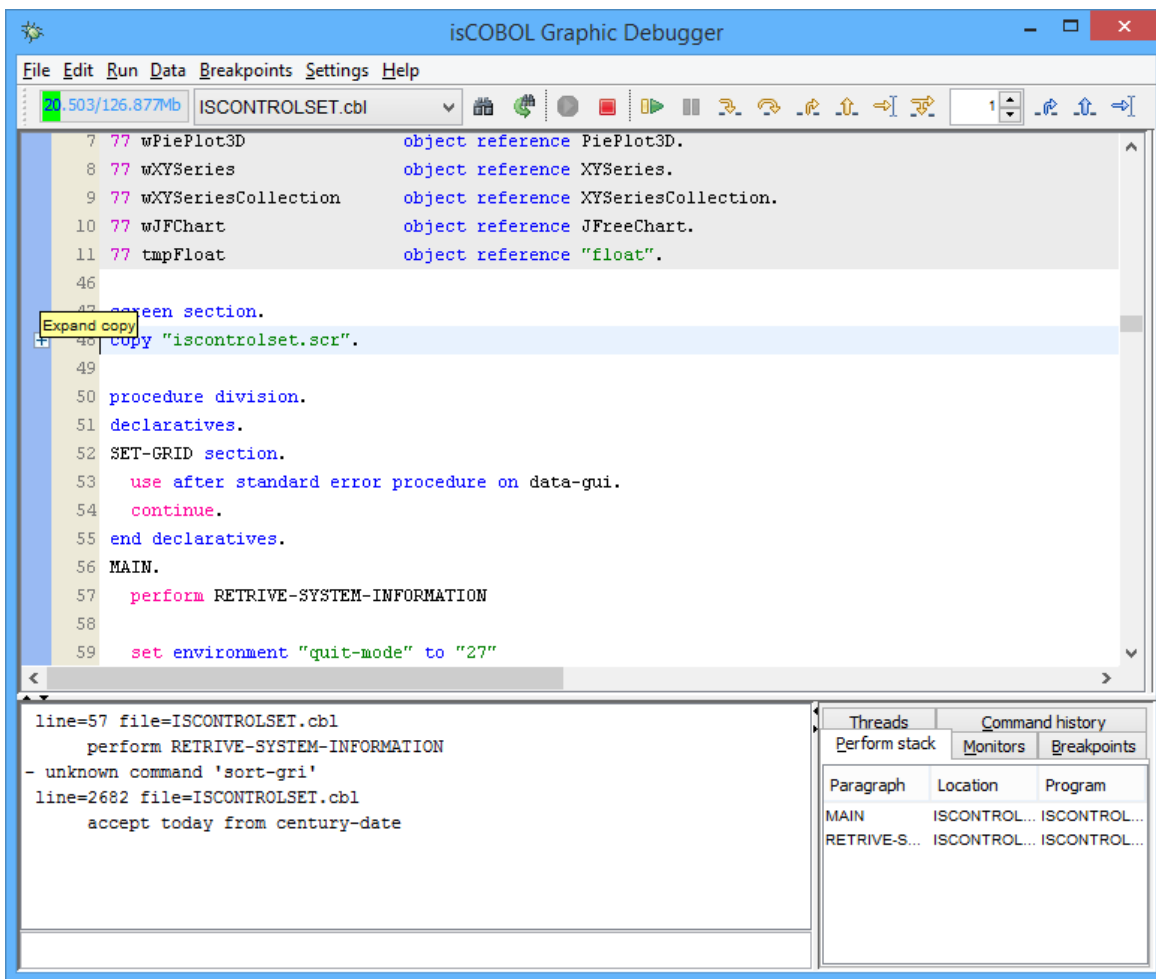
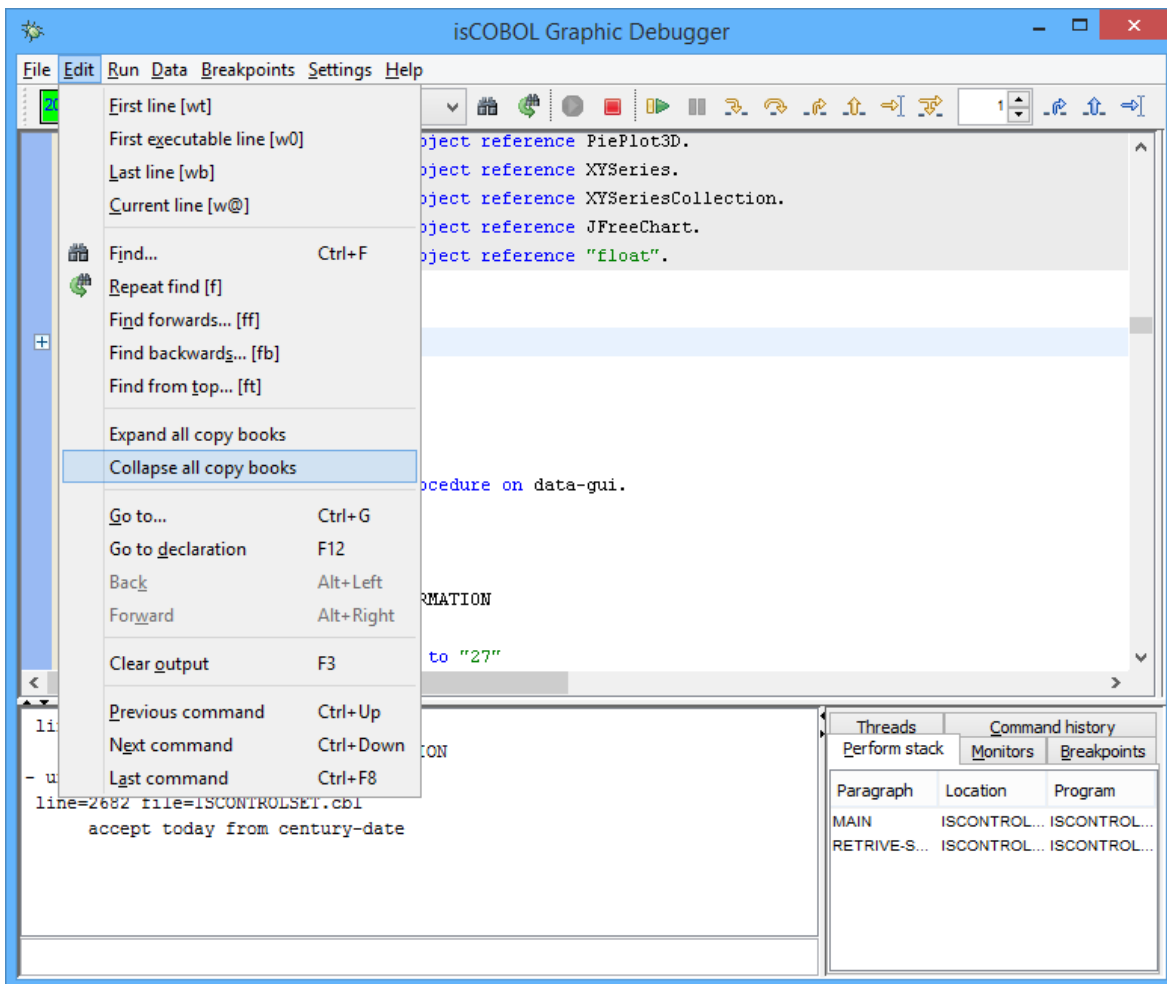


Figure 7, *New Expand and Collapse ALL*, shows (it is also possible from debugger menu) it can expand or collapse all copy files of the whole COBOL source that are going to be debugged.

**Figure 7.** New Expand and Collapse ALL

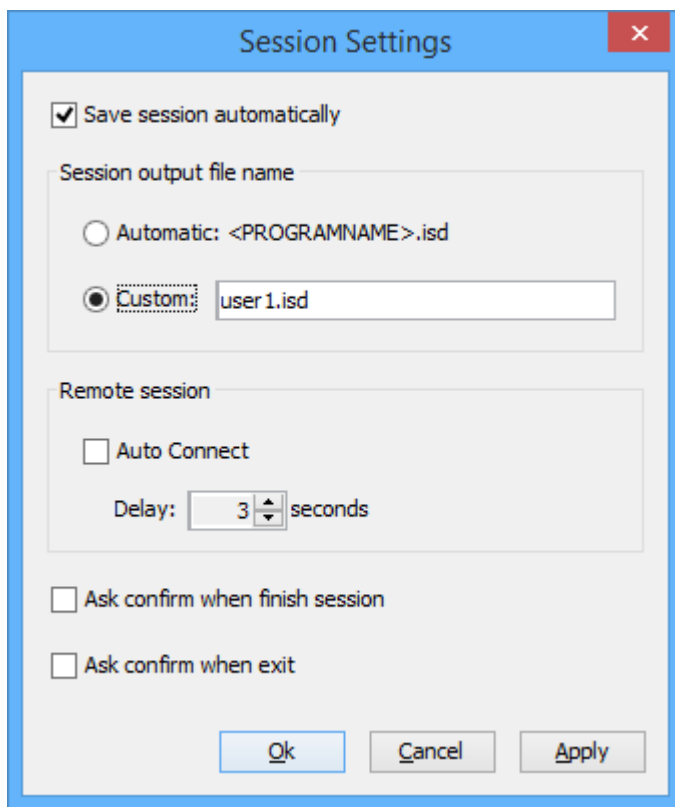




### Ability to customize the .isd file of Debugger in Session Settings

During a debugger session all useful settings like Monitors, Breakpoints and others are saved on a debugger file session in order to be restored when the debugger is started again. By default the name of the file where a session is saved take the name of the debugged COBOL program. As shown in Figure 8, *Debugger Session Setting*, a debugger user can customize the name of debugger session file.

**Figure 8.** Debugger Session Setting



Other minor enhancements

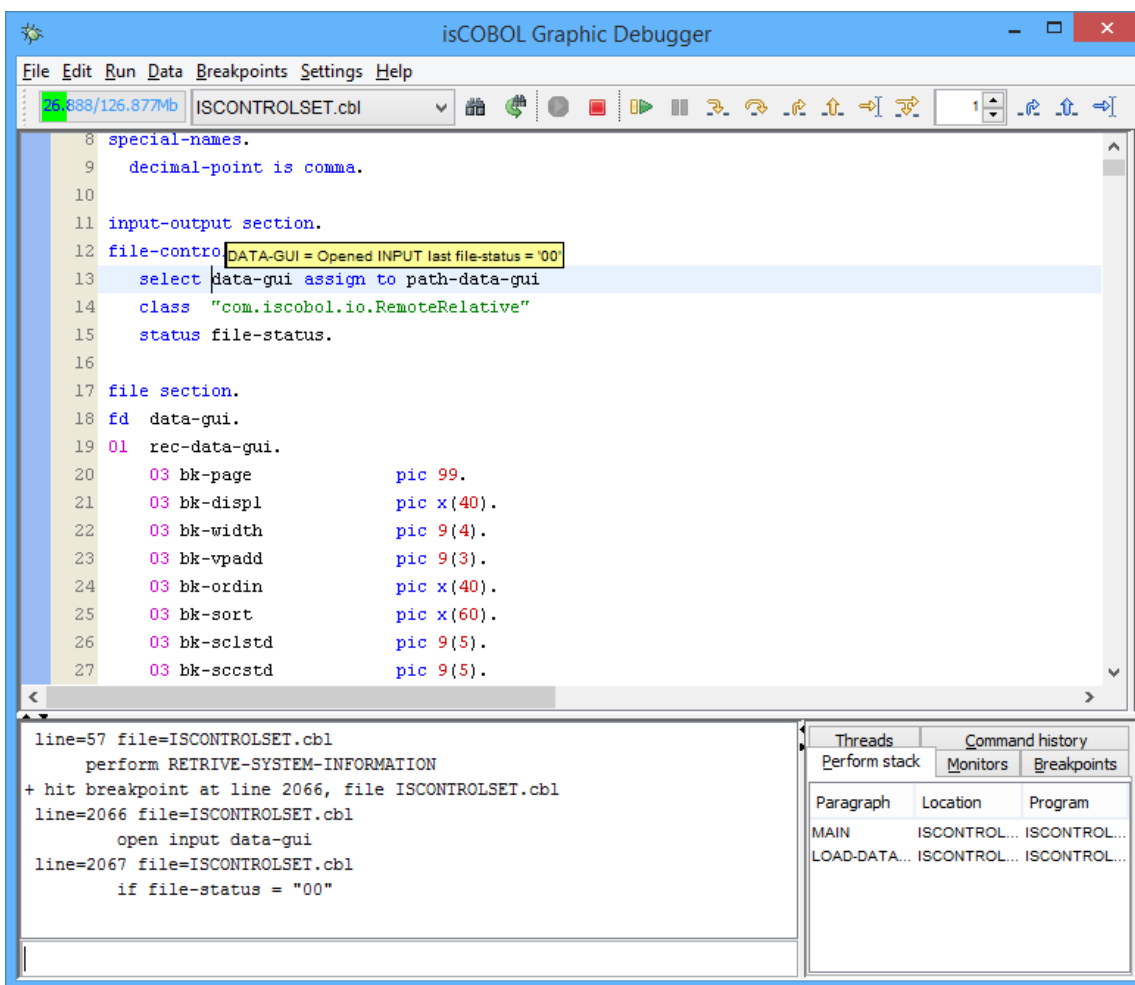
On the INDEX, RELATIVE and SEQUENTIAL file is possible to inquire if a file is opened or closed and view the last file-status. This may be done with the following command:

DISP FILE1

+ FILE1 = Opened INPUT last file-status = '00'

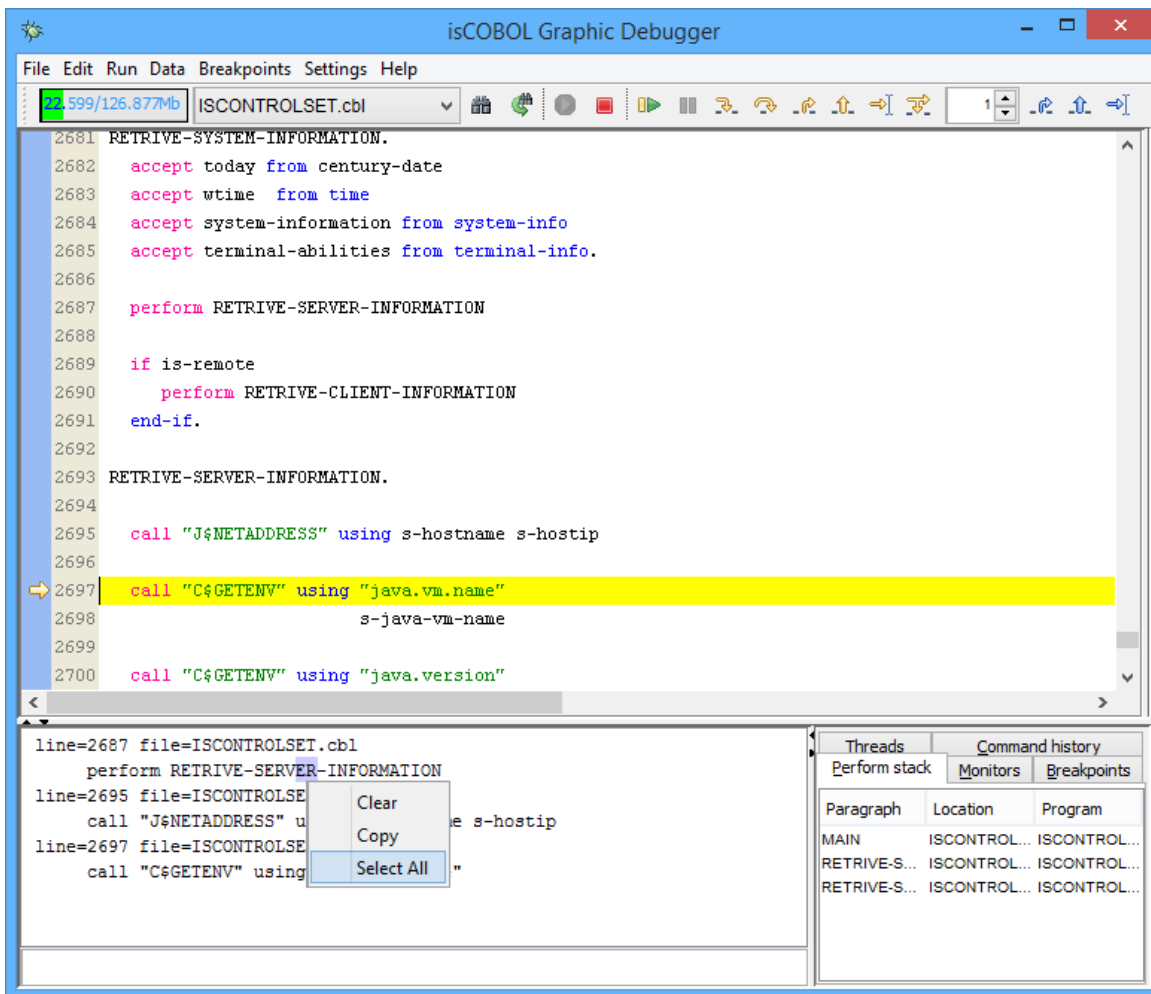
Or, as showed in Figure 9, COBOL file states, iteratively:

**Figure 9.** COBOL file states



As depicted in Figure 10, *Clear and Select all*, two additional items are available in the pop-up menu of the output window in the Debugger to easily maintain it clean and copy the output:

**Figure 10.** Clear and Select all



### **isCOBOL Database Bridge Improvements**

The isCOBOL Database Bridge tool enabling COBOL programs to use the RDBMS power without changing COBOL source code or learning ESQL was enhanced to better satisfy user's requests.

#### Improved performance

The EDBI routines now use a faster way to bind COBOL fields to DBMS fields. The performance improvement is related to the number of File Description fields. More fields are used while also increasing the performance. Up to 10% of the EDBI RDBMS subroutines will be benefit with the enhanced performance. This optimization affects only FDs with USAGE DISPLAY fields.

In EDBI routines for PostgreSQL RDBMS, NUMERIC fields are now used to bind numeric COBOL data items. NUMERIC fields perform better than INTEGER, SMALLINT and other numeric field types used by previous versions.

Improved the support for EFD WHEN directive

In previous versions EFD WHEN without TABLENAME clauses used to manage REDEFINES fields within a record definition, caused duplicated data. Now data is stored only in the field where the WHEN directive is specified. Consider the following FD:

```

01  arc-rec.
    03  arc-k  pic 999.
    03  arc-ty pic 9.
    $efd when arc-ty = 0
    03  arc-d  pic x(10).
    $efd when arc-ty = 1
    03  arc-r  redefines arc-d.
        05  arc-r1 pic x(5).
        05  arc-r2 pic x(5).
    03  arc-d2  pic x(10).

```

With previous versions, records were filled as follows:

ARC_K	ARC_TY	ARC_D	ARC_R1	ARC_R2	ARC_D2
1	0	XXXXXXXXXX			ZZZZZZZZZZ
2	1	XXXXXXXXXX	r1	r2	ZZZZZZZZZZ
3	0	YYYYYYYYYY	r1	r2	ZZZZZZZZZZ
4	1	YYYYYYYYYY	r11	r22	ZZZZZZZZZZ

Now they're filled as follows:

ARC_K	ARC_TY	ARC_D	ARC_R1	ARC_R2	ARC_D2
1	0	XXXXXXXXXX			ZZZZZZZZZZ
2	1		r1	r2	ZZZZZZZZZZ
3	0	YYYYYYYYYY			ZZZZZZZZZZ
4	1		r11	r22	ZZZZZZZZZZ

In addition AND / OR clauses are now fully supported with EFD WHEN directives.

### Support for COBOL program compiler with -cp

The isCOBOL programs that need to use the C memory model to share pointers between C and COBOL, now can also take advantage of EDBI routines.

### New property iscobol.easydb.replacement\_rules

The new configuration property `iscobol.easydb.replacement_rules=n` allows to customize how the table is named. This can be used to obtain better table names in the RDBMS when migrating from indexed files.

For example having this SELECT:

```
select file2
      assign to "CUSTOMERS.DAT"
      organization indexed ...
```

By default the table name used by isCOBOL Database Bridge would be `CUSTOMERS_DAT`, but setting:

```
iscobol.easydb.replacement_rules=4
```

Where 4 means to remove the extension, the table name becomes `CUSTOMERS`

## New User Interface Features

### New GUI control RIBBON

The isCOBOL 2015 R1 introduces a new graphical control named RIBBON. The ribbon is a command bar that organizes the features of an application into a series of tabs at the top of the application window. The ribbon increases discoverability of features and functions, that enable to learn the application quicker and makes users feel more in control.

Such ribbons use tabs to expose different sets of controls, eliminating the need for many parallel toolbars and menu bars.

Code example:

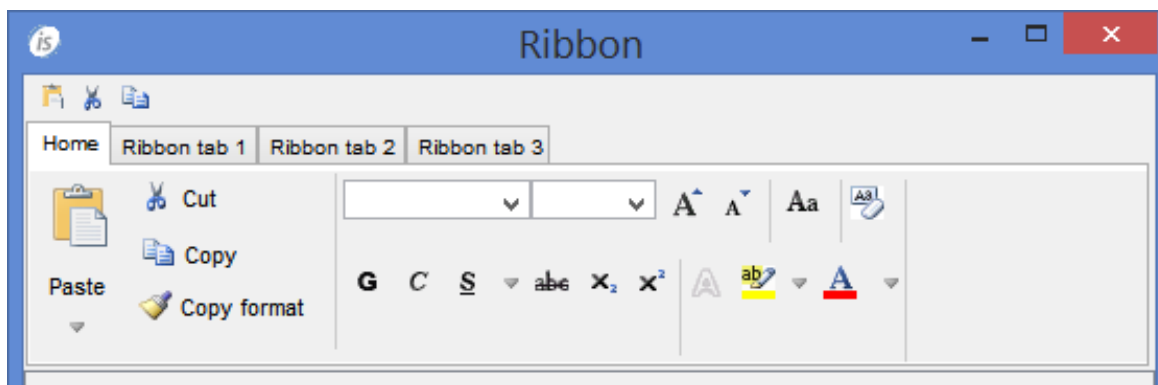
```
display ribbon
  lines 7
  handle hRibbon
  upon hWin
  HEADER-ALIGN 1
modify hRibbon, tab-to-add ("Home"
                           "Ribbon tab 1",
                           "Ribbon tab 2",
                           "Ribbon tab 3")

display Ribbon-page-1 upon hRibbon(1)
display Ribbon-page-2 upon hRibbon(2)
display Ribbon-page-3 upon hRibbon(3)
display Ribbon-page-3 upon hRibbon(4)
```

Every control inside the screens (Ribbon-page-1, Ribbon-page-2, ...) can have the new style ON-HEADER in the first line with a small button.

Figure 11, *Ribbon Control*, shows the final result with the ON-HEADER style set on the buttons Paste, Cut and Copy.

**Figure 11.** Ribbon Control



### TAB controls can act like real TAB folder container

New properties in the TAB-CONTROL are available to allow multiple screens to be contained within a single container. With this new approach to design TAB elements, it is able to redisplay the screen for every navigational TAB. This simplifies the code to manage the TAB folding and also improves the network performances because the tab navigation is executed on client side.

The following example shows how to use the new style and properties ALLOW-CONTAINER, TAB-GROUP, TAB-GROUP-VALUE:

```
01 screen1.
  03 entry-field line 2 col 20 size 10.
  03 tab1 tab-control line 3 col 2 size 70 lines 10
      ALLOW-CONTAINER
      value tab1-page.
  03 page1
      TAB-GROUP tab1
      TAB-GROUP-VALUE 1.
      05 label line 5 col 6 title "pag1".
      05 entry-field line 8 col 6 size 5.
  03 page2
      TAB-GROUP tab1
      TAB-GROUP-VALUE 2.
      05 label line 5 col 26 title "pag2".
      05 entry-field line 8 col 26 size 5.
```

### Programmatically use of TCP/IP packet optimizer

The Client/Server TCP/IP communication of isCOBOL Server is automatically optimized in order to reduce the number of TCP/IP packets for each GUI statement that is shown on the screen. To provide more flexibility to the developers, some op-codes were implemented in W\$FLUSH library routine to inhibit the flush and restore the flush between server and client. This allows you to improve performances in case more INQUIRE statements are executed. Using W\$FLUSH can reduce the number of packets that need to be sent over the network.

Code example:

```
call "W$FLUSH" using INHIBIT-FLUSH win-handle
inquire ef1 value in v1
inquire ef2 value in v2
perform varying ind from 1 by 1 until ind > 6
  inquire h-ef(ind) value in var(ind)
end-perform
perform varying ind from 1 by 1 until ind > grid-last-row
  inquire grid1(ind, 1) value in gr-col1(ind)
end-perform
call "W$FLUSH" using ALLOW-FLUSH win-handle
```



### Common property CUSTOM-DATA

To extend the concept of the HIDDEN-DATA property to all GUI controls, a new common property named CUSTOM-DATA has been implemented. It allows isCOBOL developers to save personal information to each GUI controls. The property can be used on DISPLAY, MODIFY and INQUIRE.

Code example:

```
display label line 2 col 2 title "Label1" custom-data 1
      handle in hlab1
inquire hlab1 custom-data in ws-custom-data
modify  hlab1 custom-data 2
```

### Support for JavaFX WebView web-browser

A new web-browser implementation has been made for the WEB-BROWSER control to take advantage on JavaFX WebView.

JavaFX WebView, available from JSE 1.7, is the first web-browser implementation written 100% in Java. It supports Cascading Style Sheets (CSS), JavaScript, Document Object Model (DOM), and HTML5. Using this option avoids using native code and reduces the overhead to invoke the Internet Explorer ActiveX.

To enable it, it's necessary to set the configuration setting:

```
iscobol.gui.webbrowser.class=com.iscobol.fx.JFXWebBrowser
```

### Adding style READONLY for DATE-ENTRY

The style READONLY is now supported on the DATE-ENTRY control to disable manual input by keyboard on the entry-field part that composes the DATE ENTRY control. It forces users to utilize the calendar control instead of typing data on the entry-field.

## Framework Improvements

### New profiling facility

To help tune application performance isCOBOL 2015 R1 includes a profiling facility during the execution of a program. This built-in facility provides an easy way to choose the COBOL program that should be profiled, prompting the runtime to collect information about the amount of time spent in different parts of the code. All this information is placed into an output file called iscobol.txt.prof.

It is very easy to profile any isCOBOL application because this profiling facility can be used on isCOBOL programs at runtime level without any need for the compiler option.

This built-in facility is activated when a program is executed with the "-javaagent" flag:

```
isrun -J-javaagent:isprofiler.jar IO_PERFORMANCE
```

That creates an iscobol.hprof.txt file containing:

```
isCOBOL profile rev $Revision: 18624 $ created Tue Apr 14 14:35:08 CEST 2015
elapsed time = 3.94
evaluated time = 2.52
overhead1=56; overhead2=75
-----+-----+-----+-----+-----
self % |accum %| seconds | count | program:paragraph
-----+-----+-----+-----+-----
25.96%| 25.96%| 0.65 | 1 | IO_INDEXED:DELETE_FILE1_TEST
16.09%| 42.06%| 0.40 | 1 | IO_INDEXED:LOAD_FILE1_TEST
14.80%| 56.86%| 0.37 | 1 | IO_INDEXED:UPDATE_FILE1_TEST
9.15%| 66.01%| 0.23 | 1 | IO_INDEXED:READ_FILE1_TEST
8.66%| 74.68%| 0.21 | 1 | IO_SEQUENTIAL:LOAD_FILE1_TEST
5.34%| 80.02%| 0.13 | 1 | IO_RELATIVE:UPDATE_FILE1_TEST
4.56%| 84.59%| 0.11 | 1 | IO_RELATIVE:DELETE_FILE1_TEST
3.22%| 87.81%| 0.08 | 1 | IO_PERFORMANCE:MAIN_LOGIC
3.20%| 91.01%| 0.08 | 1 | IO_RELATIVE:LOAD_FILE1_TEST
3.12%| 94.14%| 0.07 | 1 | IO_SEQUENTIAL:READ_FILE1_TEST
2.73%| 96.88%| 0.06 | 1 | IO_RELATIVE:READ_FILE1_TEST
1.69%| 98.57%| 0.04 | 1 | IO_INDEXED:MAIN_LOGIC
0.52%| 99.10%| 0.01 | 4 | IO_INDEXED:START_TIMER
0.40%| 99.50%| 0.01 | 4 | IO_RELATIVE:START_TIMER
0.37%| 99.88%| 0.00 | 2 | IO_SEQUENTIAL:START_TIMER
0.03%| 99.92%| 0.00 | 1 | IO_RELATIVE:MAIN_LOGIC
0.03%| 99.96%| 0.00 | 1 | IO_SEQUENTIAL:MAIN_LOGIC
0.01%| 99.98%| 0.00 | 4 | IO_INDEXED:STOP_TIMER
0.01%| 99.99%| 0.00 | 4 | IO_RELATIVE:STOP_TIMER
0.00%| 100.00%| 0.00 | 2 | IO_SEQUENTIAL:STOP_TIMER
```

### Enhanced Win\$Printer and Print functionalities

To simplify the background color definition, a new op-code named WINPRINT-SET-BACKGROUND-COLOR was provided. It sets the background-color to be used during printing.

Code example:

```
initialize wprtdata-text-color
move wprt-color-yellow to wprtdata-text-color
call "win$printer" using WINPRINT-SET-BACKGROUND-COLOR
                        wprtdata-text-color
```

In order to renovate the printing output with some useful information like page-number, name of report, date of printing, etc, a new op-code WINPRINT-SET-HEADER-FOOTER for WIN\$PRINTER was provided. The following example shows how to have the date printed on the header and the number of page printed on the footer:

```
call "win$printer" using WINPRINT-SET-HEADER-FOOTER
                        "&b Printed on date &D"
                        "&b Page &p of &P"
                        h-large-font
```

To be able to generate an encrypted PDF file, new attributes are now supported in WIN\$PRINTER library routine to manage passwords:

Code example:

```
initialize pdfcrypt-type
add pdfcrypt-std-128 pdfcrypt-all-permissions
   giving pdfcrypt-type
call "win$printer" using winprint-set-attribute
   "ENCRYPTION" pdfcrypt-type
call "win$printer" using winprint-set-attribute
   "USER_PASSWORD" "MyPwd"
```

## Other enhancements

Here some minor enhancements provided in the isCOBOL 2015R1 framework:

- Ability to set variable value inside properties file

```
iscobol.conf.var_delimiters=${.}  
iscobol.myenv=from Java  
iscobol.hello=hello ${iscobol.myenv} ${java.version}
```

- Ability to dynamically set the fetch size for an ESQL cursor statement.

A new property named `iscobol.jdbc.fetch_size` allows you to specify the number of rows to retrieve in one trip from the database. This can be used in ESQL programs or with isCOBOL Database Bridge.

- I\$IO info-function to return collating sequence.

This new feature is also applied to JUTIL and ISMIGRATE utilities when working with indexed files with collating sequence.

- Special convention to specify DLL calling convention during CALL statements.

To force all functions of mylib with C convention:

```
CALL "mylib.dll/0"
```

To force all functions of mylib with PASCAL convention:

```
CALL "mylib.dll/1"
```

## **isCOBOL Server Improvements**

Starting from isCOBOL 2015R1, isCOBOL Server was improved to provide more functionality, better performance and less use of memory.

### Encrypted SSL connections

SSL is an industry standard used by millions of websites to protect the online transactions with their customers. Users that need all data to be passed between the isCOBOL Server and isCOBOL Thin Client remain private and integral as a result of this SSL standard support.

Additional isCOBOL properties are provided to specify keystore, keystore password, truststore and truststore password.

On the server site:

```
iscobol.net.ssl.key_store=path  
iscobol.net.ssl.key_store_password=pwd
```

On client site:

```
iscobol.net.ssl.trust_store=path or *  
iscobol.net.ssl.trust_store_password=pwd
```

### Enhanced configuration settings

It is possible to define a new property named `iscobol.as.password_file` to set a customized `password_file`. This allows using the same password files from different Application Servers started on different folders and running on the same server:

```
iscobol.as.password_file=/etc/mypwd.properties
```

### Enhanced internal lock manager

The property `iscobol.file.index.autolock_allowed` is now supported also in Application Server when running with the `InternalLockManager`.

### Optimized network performance

On high latency networks, where the round-trip delay time is high, it is important to try to reduce the total number of packets used. Starting from isCOBOL 2015R1, many areas were improved in the Application Server traffic in order to increase screen performances.

For example on the DISPLAY screen-section and ACCEPT statements we used 30% less number of packets. Also the INQUIRE statement, that usually breaks any optimizer option that forces communication, was optimized avoiding to ask the property value on the client side where possible.

Many TCP-IP round-trips were removed at startup time where it is necessary to send from the server to the client all configuration settings. This optimization reduces the time spent when the isCOBOL Thin Client goes live.

## isCOBOL EIS improvements

### Simplified the migration from legacy CGI programs

The IS EXTERNAL-FORM clause, which associates a group item with HyperText Markup Language (HTML) data using the Common Gateway Interface (CGI) specification, is now supported.

```
01 MY-FORM IS EXTERNAL-FORM.  
03 CGI-VAR1 PIC X(20) IDENTIFIED BY "Name".  
03 CGI-VAR2 PIC X(50) IDENTIFIED BY CGI-VAR1 .
```

isCOBOL already provided a more powerful syntax but having EXTERNAL-FORM clause could dramatically speed-up migration of existing CGI COBOL programs to isCOBOL.

### Embedded HTML support

Embedded HTML is another option to enable you to output HTML directly from a COBOL CGI program. Using Embedded HTML, you can output HTML statements, complete or partial HTML pages, or a combination of both, without including any special data declarations. HTML files that contain substitution markers can be merged with COBOL variable data.

```
EXEC HTML  
    Customer Name :cust-name <BR>  
END-EXEC
```

Embedded HTML support is enabled in isCOBOL compiler turned on the HTML pre-compiler with a special option -exec=html.

Support of Embedded HTML could dramatically speed-up migration of existing CGI COBOL programs to isCOBOL.

### Enhanced HTTP classes

HTTPHandler class has been enhanced, adding the ability to process html strings with the new method:

```
processHtmlString ().
```

This method is similar to the existing processHtmlFile() method, the difference is that the HTML code is contained in the alphanumeric variable txt instead of in the external file.

A var must be an XML variable whose names correspond to the embedded values in the string. Example:

```
01 params identified by "_".
02 identified by "Counter".
03 Counter pic z(8)9.
...
Ink-area:>processHtmlString (
  "<H1>This page has been accessed %%Counter%% times.</H1>"
  params)
```

Enhanced HTTPClient class to post Multipart and download binary file with the new methods:

```
doPostMultipart (siteAddress, parameters).
```

```
saveResponseRaw (fileName).
```

The first method enables analogous to doPost, however it sends the parameters using the multipart/form-data protocol.

The second method allows saving the response received from the web server in the specified binary file. Example:

```
httpclient:>saveResponseRaw (fileName).
```

Enhanced HTTPData.Params class to add files in parameters with the new methods:

```
addFile (fieldName, fileName, mimeType).
```

```
addFile (fieldName, fileName).
```

These methods allow using the previous doPostMultipart method in the HTTPClient class.

The latter set the mime type of the file as "application/octet-stream". Example:

```
set pars = http-params:>new
  :>add ("param1", 1)
  :>addFile ("upfile_0", "myfile.txt")
set httpclient = http-client:>new()
httpclient:>doPostMultipart (
  "http://www.mywebsite.com/index.php?Section=n pars
  ).
```

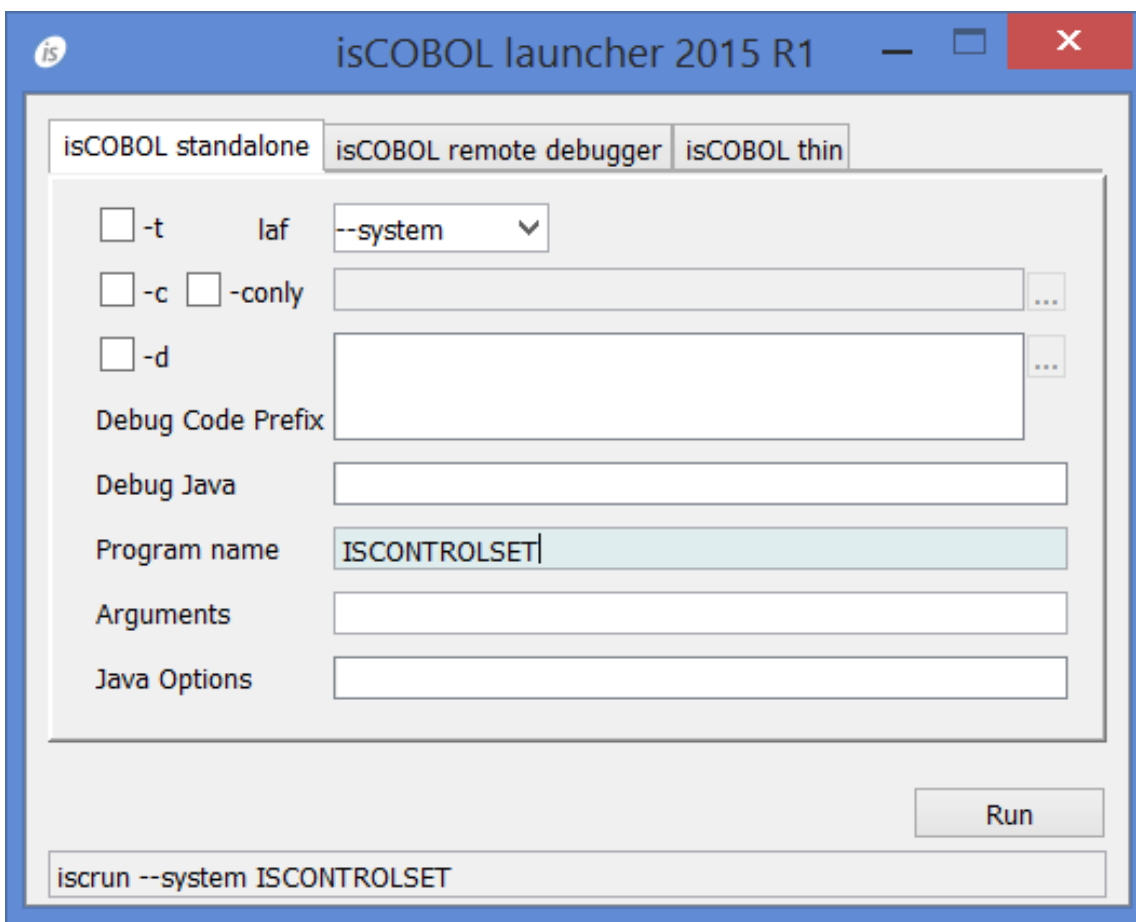


### Utility improvements

#### New utility ISL "isCOBOL Launcher"

ISL provides an easy and quick way to run an isCOBOL program without knowing the options and properties that will be used. Looking at Figure 12, *isCOBOL Launcher*, it is clear that ISL could help new isCOBOL users to understand how many ways they can interact with isCOBOL objects. On the status bar of the utility, new users can start to learn the needed options.

**Figure 12.** IsCOBOL Launcher

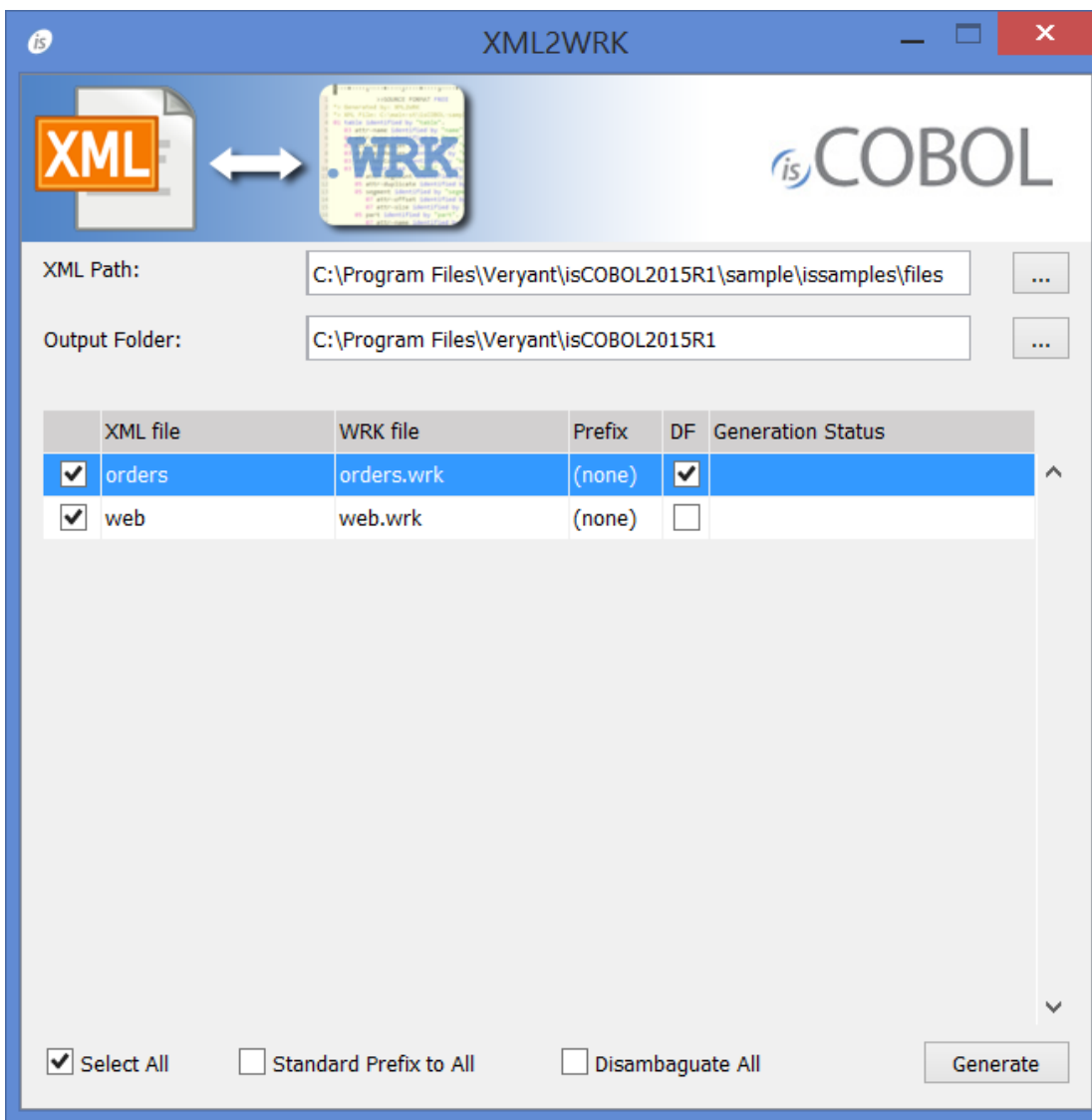


New GUI interface for XML2WRK utility

The XML2WRK utility that opens an XML file and creates the corresponding record definition used with the XMLStream Class (com.iscobol.rts.XMLStream) object, is now released with a graphical user interface improving its usage.

As depicted in Figure 13, *XML2WRK Utility*, in addition to the new GUI interface, it is also possible to parse remote xml files via URL and generates namespaces.

**Figure 13.** XML2WRK Utility

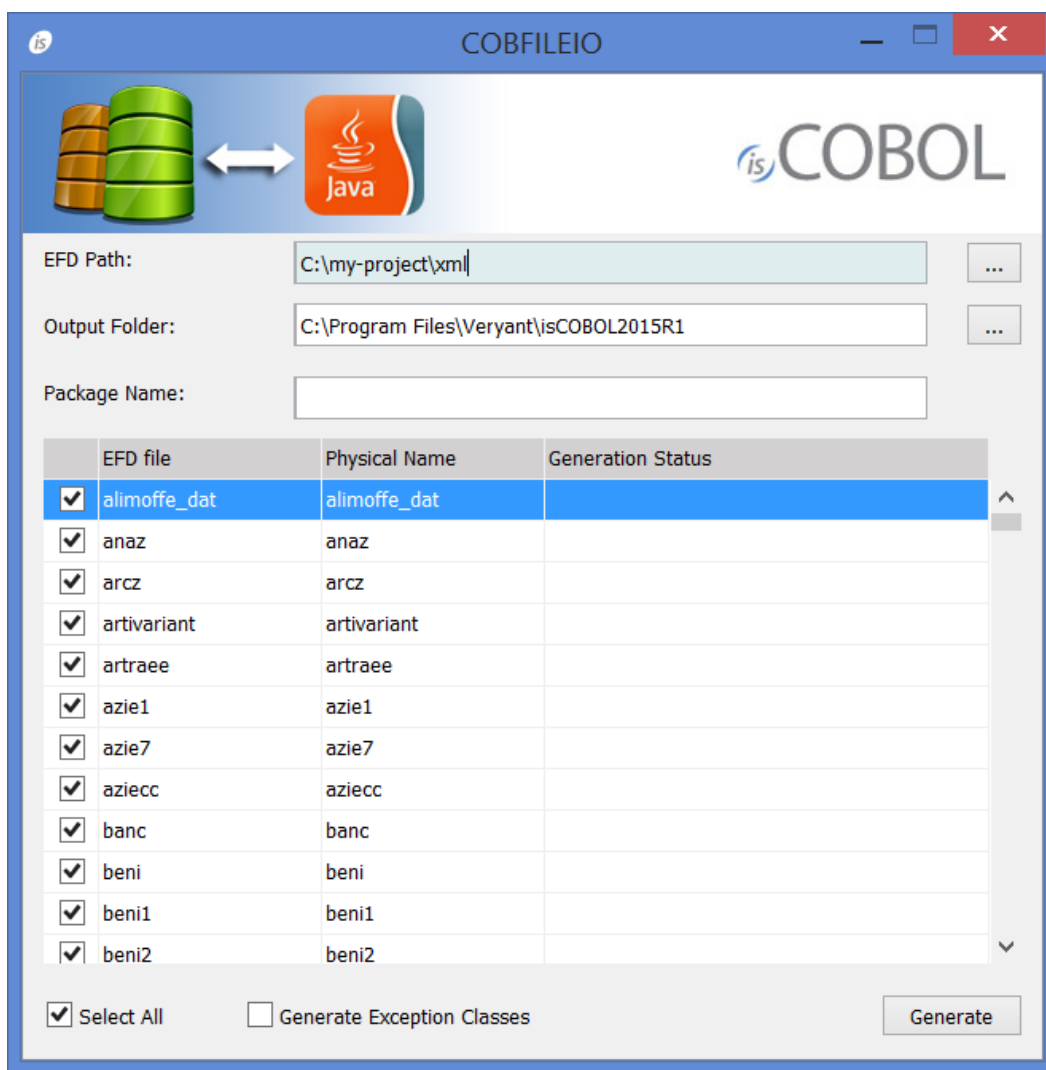


New GUI interface for COBFILEIO utility

COBFILEIO, the utility that generates Java classes used to access COBOL ISAM files and records from Java programmer, is now released with a graphical user interface. As depicted in Figure 14, *COBOLFILEIO utility*, this new GUI will make easier to be used for all users.

The Java programmer does not need any knowledge of COBOL data types or their underlying storage format, and COBFILEIO automatically generates Javadocs for the files and record classes.

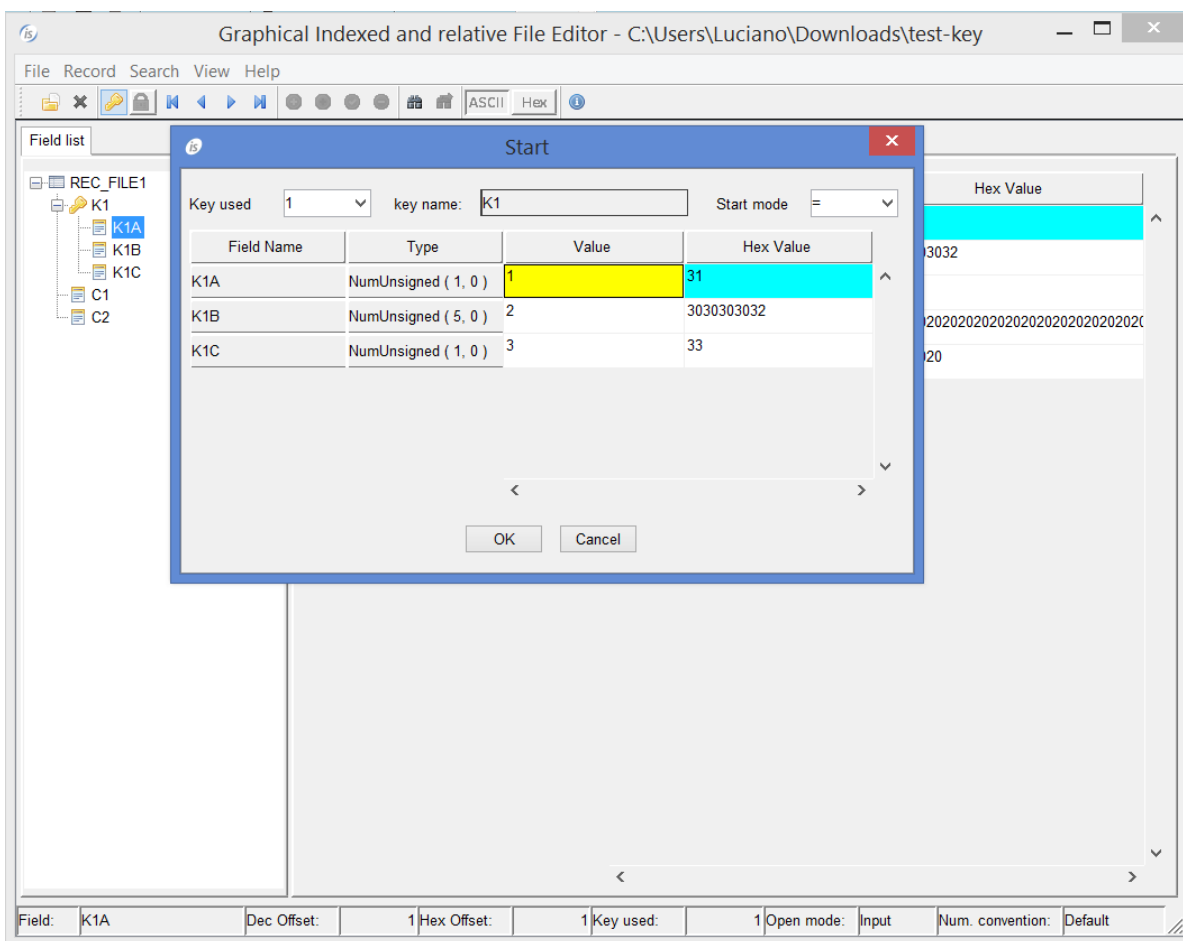
**Figure 14.** COBOLFILEIO utility



**Graphical Indexed and relative File Editor improvement**

When the EFD file is provided, the field info is used during START instead of having just one entry-field that allows filling the key value. Now a grid is used in the interface where every line allows setting the field "Value" of each segment that composes the key. Figure 15, *Graphical Indexed and relative File Editor*, shows this new feature.

**Figure 15.** Graphical Indexed and relative File Editor



ISMIGRATE improvements

ISMIGRATE, the utility that converts ISAM files from one to another format now supports DBMaker DBMS as standard option. As depicted in Figure 16, *DBMaker DCI support*, a new page was added to provide the database name, user and password. The data migration is executed using DBMaker DCI interface.

**Figure 16.** DBMaker DCI support

