isCOBOL^{TM} Evolve

isCOBOL Evolve 2017 Release 2 Overview

isCOBOL Evolve 2017 Release 2 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2017 R2.

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

The isCOBOL IDE now supports the use of an external compiler instead of the one shipped with the IDE, allowing developers to use the latest IDE while building projects with older compiler versions, or to use an older IDE to compile projects using later versions of the compiler.

The “Reload all Linked Copies” feature allows one-click refresh of all linked copy file in the Screen Section, Working Storage and Linkage Section editors in the IDE.

The Database Bridge has been rewritten in Java, and is now integrated in the compiler and available on all supported platforms.

MDI (Multiple Document Interface) windows are now supported to enhance COBOL applications GUIs.

isCOBOL EIS has been enhanced for better REST web service creation, allowing automatic detection of data formats, and WebDirect 2.0 now supports true web-style paginated grids.

Details on these enhancements and updates are included below.

isCOBOL IDE Enhancements

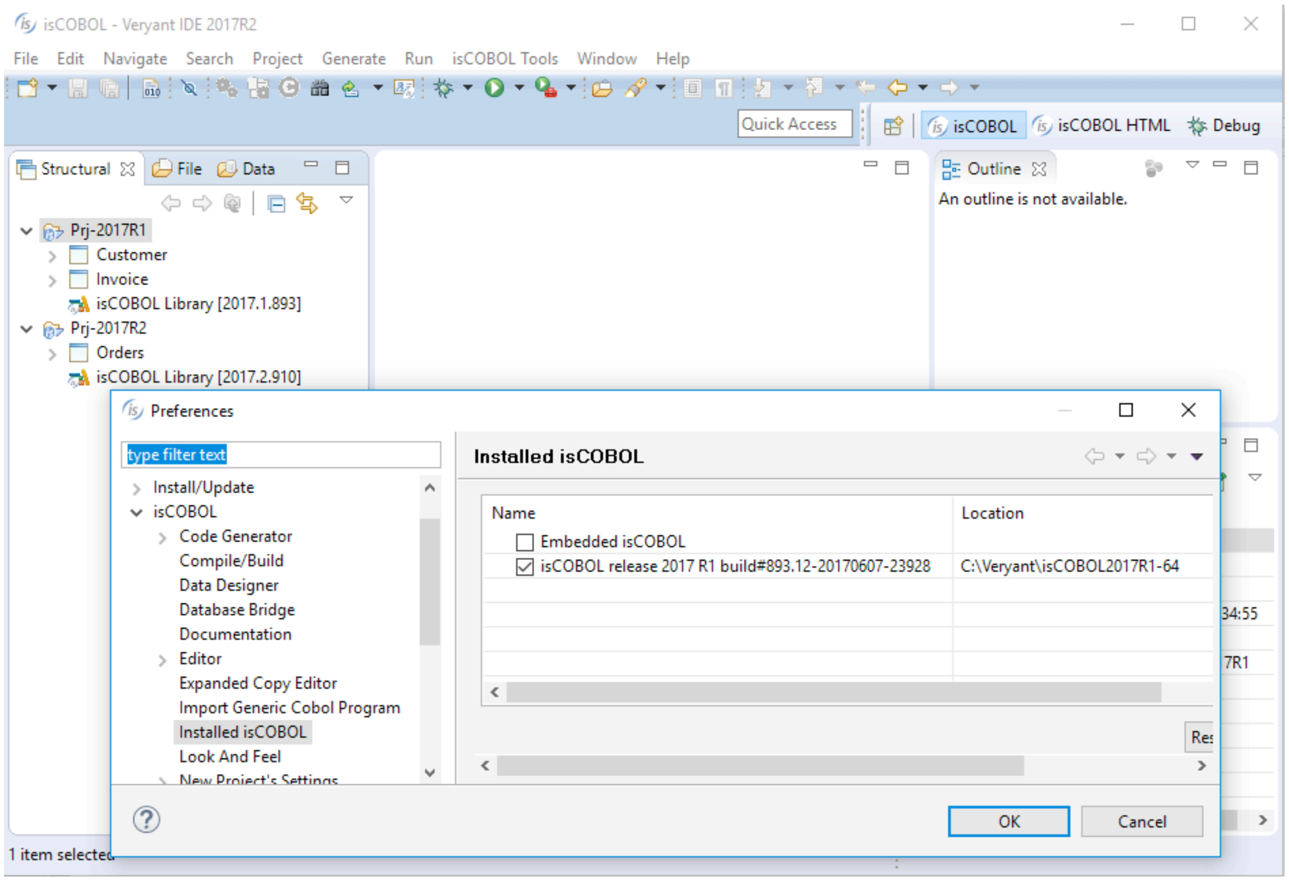
The isCOBOL 2017 R2 IDE now allows developers to set a different isCOBOL compiler version to use while compiling and testing for each project in the workspace, instead of the one contained within the IDE.

This enables the use of the latest isCOBOL IDE's features, while still compiling and running programs using different isCOBOL compiler versions, simplifying development and testing of applications already in production with any isCOBOL version compatible with this feature.

The minimum compatible compiler version is 2017 R1.

The Figure 1, *Using multiple isCOBOL versions*, shows two projects in the same workspace configured to use different isCOBOL versions.

Figure 1. Using multiple isCOBOL versions



An outline is not available.

- Prj-2017R1
  - Customer
  - Invoice
  - isCOBOL Library [2017.1.893]
- Prj-2017R2
  - Orders
  - isCOBOL Library [2017.2.910]

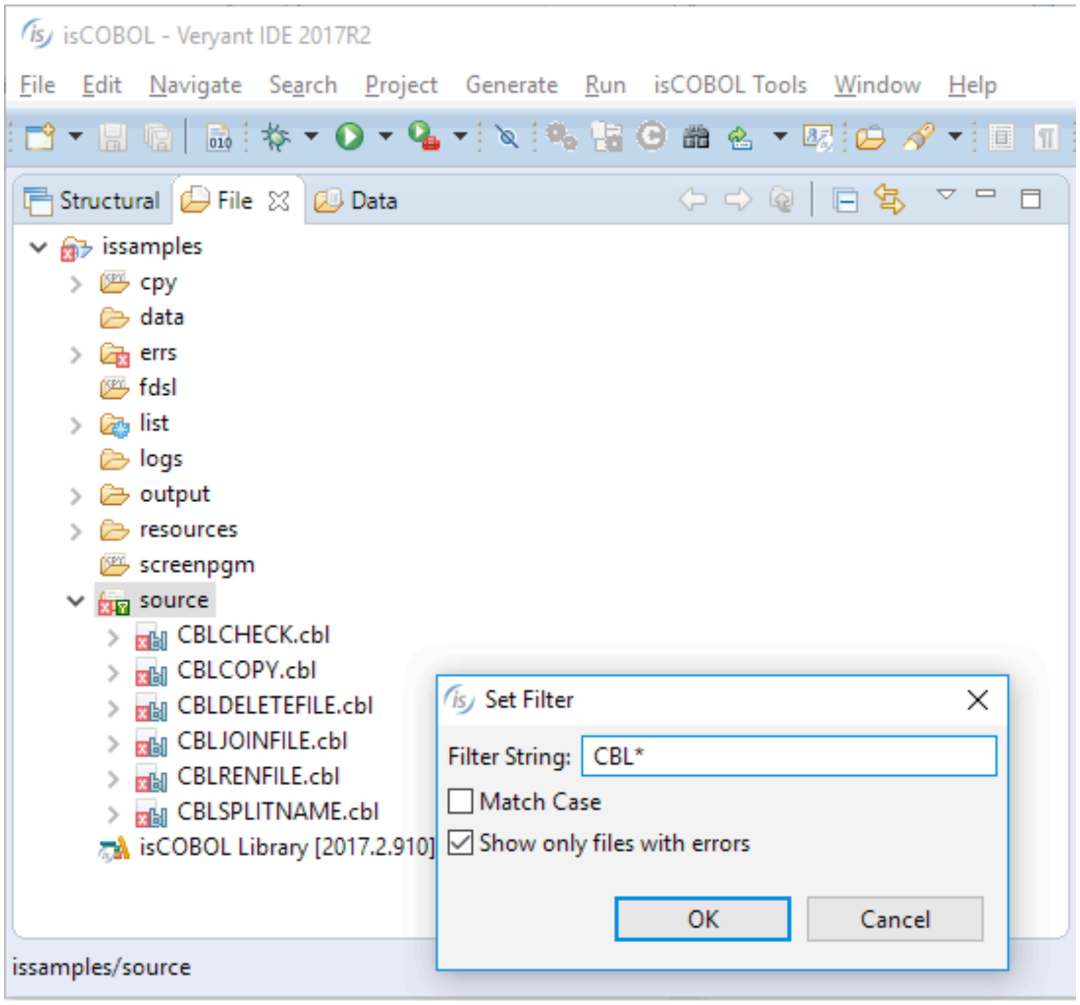
1 item selected

Installed isCOBOL

Name	Location
☐ Embedded isCOBOL	
☑ isCOBOL release 2017 R1 build#893.12-20170607-23928	C:\Veryant\isCOBOL2017R1-64

The "Set Filter" feature has been improved by adding the option to show only programs that have compilation errors, making it much easier to find source files that need fixing in very large projects, as shown in Figure 2, Set filter panel.

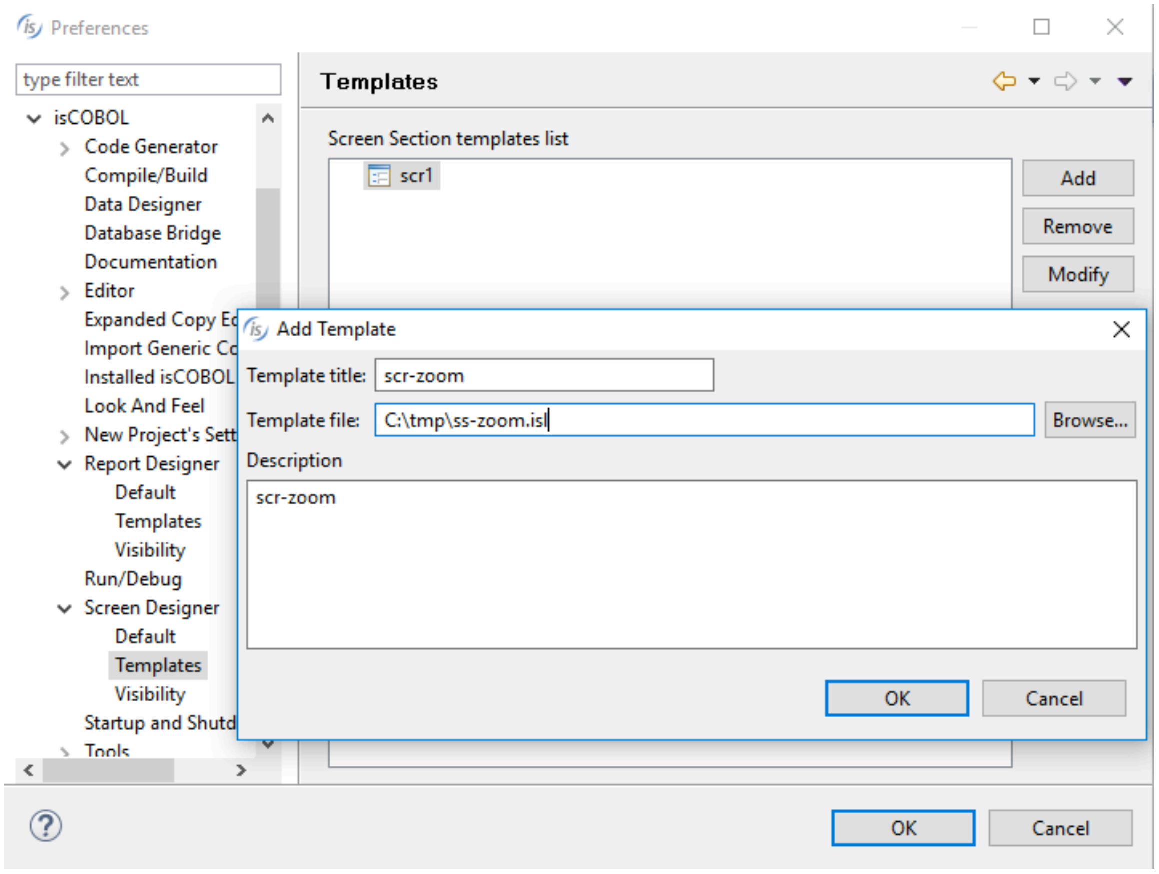
Figure 2. Set filter panel



Filter String: CBL*
[ ] Match Case
[✓] Show only files with errors

The import Report and Screen features have been improved by allowing the use of custom templates, and by allowing the Report and Screen prefix to be modified when importing the template, enabling teams to set standards when creating new screens and reports in an application. Templates can be set in the “Preferences / isCOBOL / Screen Designer / Templates” and “Preferences / isCOBOL / Report Designer / Templates” pages, as shown in Figure 3, *Templates definition*.

Figure 3. Templates definition



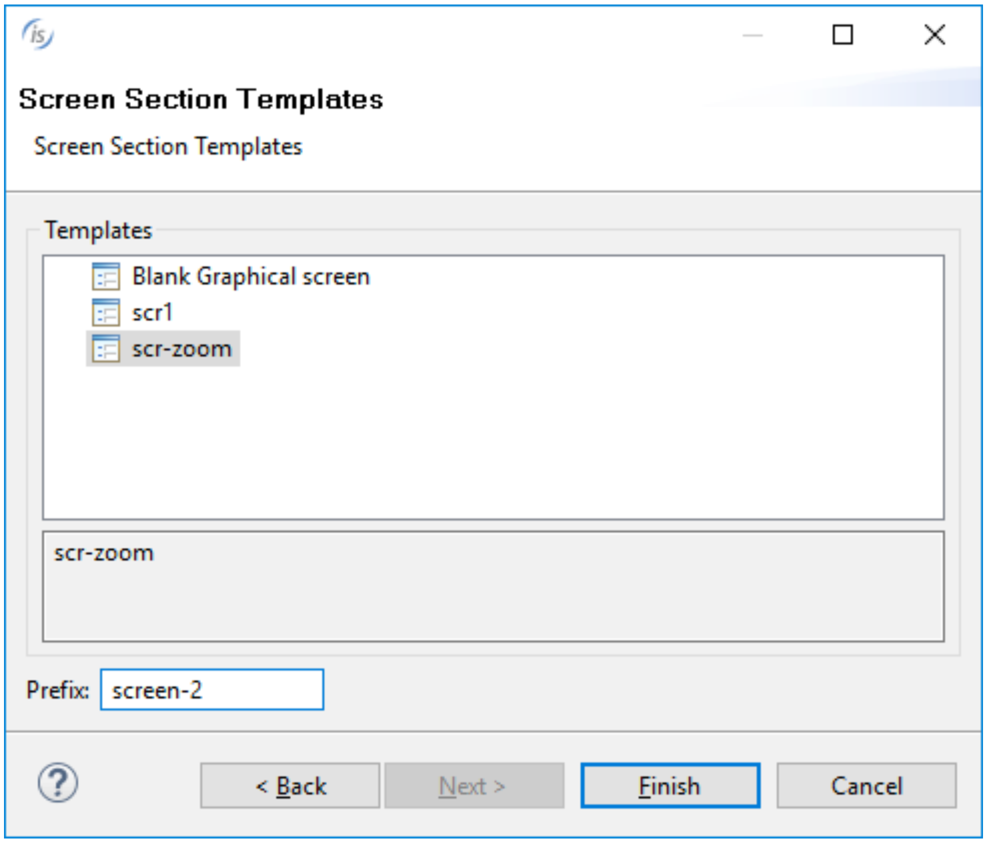
Template title: scr-zoom

Template file: C:\tmp\ss-zoom.isl

Description: scr-zoom

When creating a new screen or report, a template can be selected and a prefix can be specified to generate unique control names, as **Figure 4**, *New Screen from custom Template* shows. The prefix can also be changed at a later time, using the Change Prefix dialog.

Figure 4. New Screen from custom Template



Screen Section Templates

Screen Section Templates

Templates

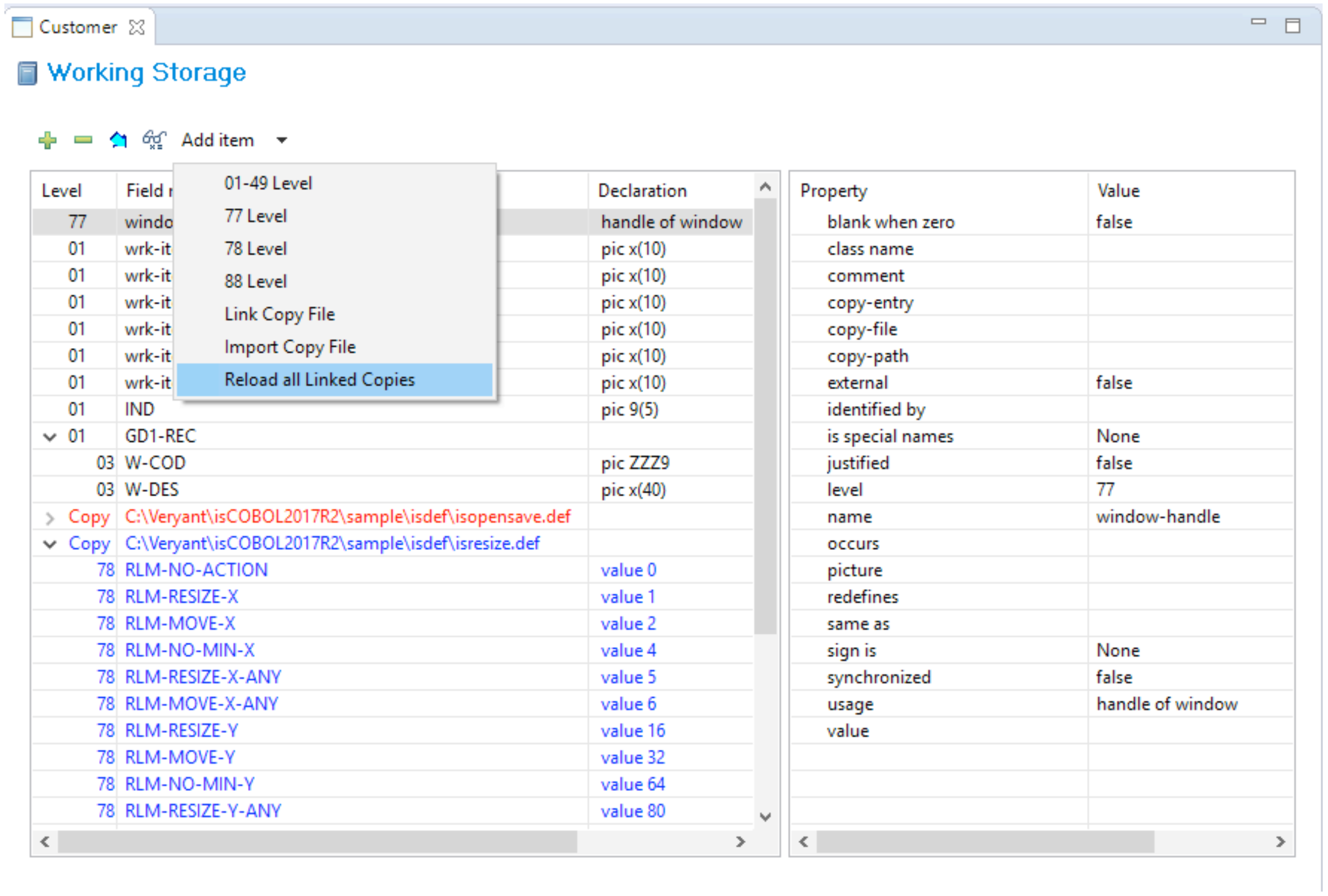
- Blank Graphical screen
- scr1
- scr-zoom

scr-zoom

Prefix: screen-2

The new "Reload all Linked copies" option is available in the Working Storage, Linkage Section and Record Definition editors, to reload all the linked copy files used within the editor in a single click. This feature can be automatically executed when starting the editors by checking the new setting in the “Data designer” option page of the IDE’s preferences, as depicted in Figure 5, *Reload when opening*.

Figure 5. Reload when opening



Level	Field r	Declaration
77	windo	handle of window
01	wrk-it	pic x(10)
01	wrk-it	pic x(10)
01	wrk-it	pic x(10)
01	wrk-it	pic x(10)
01	wrk-it	pic x(10)
01	wrk-it	pic x(10)
01	IND	pic 9(5)
01	GD1-REC	
03	W-COD	pic ZZZ9
03	W-DES	pic x(40)
Copy	C:\Veryant\isCOBOL2017R2\sample\isdef\isopensave.def	
Copy	C:\Veryant\isCOBOL2017R2\sample\isdef\isresize.def	
78	RLM-NO-ACTION	value 0
78	RLM-RESIZE-X	value 1
78	RLM-MOVE-X	value 2
78	RLM-NO-MIN-X	value 4
78	RLM-RESIZE-X-ANY	value 5
78	RLM-MOVE-X-ANY	value 6
78	RLM-RESIZE-Y	value 16
78	RLM-MOVE-Y	value 32
78	RLM-NO-MIN-Y	value 64
78	RLM-RESIZE-Y-ANY	value 80

Property	Value
blank when zero	false
class name	
comment	
copy-entry	
copy-file	
copy-path	
external	false
identified by	
is special names	None
justified	false
level	77
name	window-handle
occurs	
picture	
redefines	
same as	
sign is	None
synchronized	false
usage	handle of window
value	

New User Interface Features

MDI (Multiple Document Interface) windows are now supported to enhance COBOL applications GUIs.

Menus and pop-up menus with many items can now be scrolled for easier navigation of complex applications.

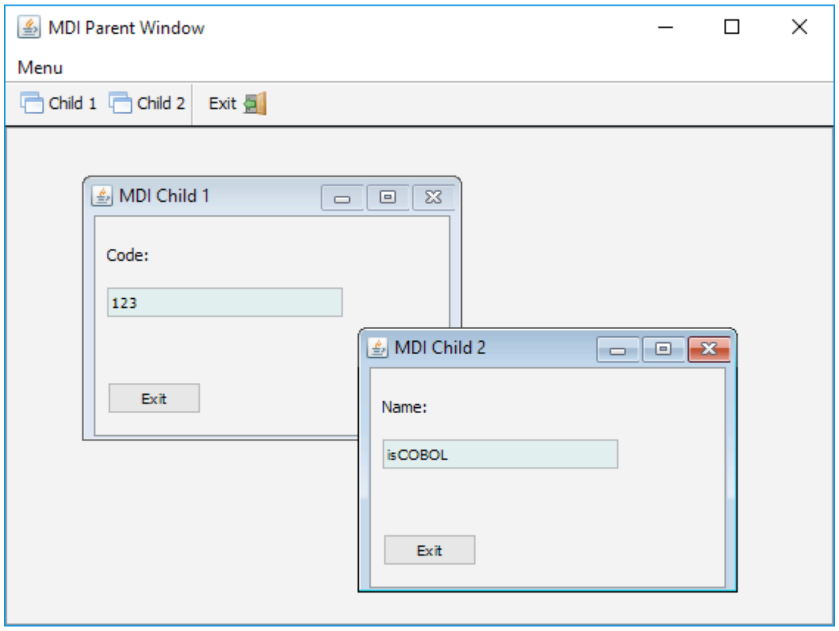
Several other minor enhancements designed to improve User Interface handling.

MDI windows

Multiple Document Interface is a user interface model to create applications that enable users to work with multiple documents (screens) at the same time. Each document runs in a separate container with its own controls for user interaction. The user can view and work on multiple documents at the same time, such as customer details, order form, and balance checking, by simply moving the cursor from one window to another.

A parent Window can contain multiple child windows, and child windows can intercept events occurring in buttons placed in the parent window tool-bar. MDI windows can be used both in single and multi-threaded environments, where several "accept" statements are executed concurrently. Figure 6, *MDI Windows* shows isCOBOL MDI Windows in action.

Figure 6. MDI Windows



Code snippet showing MDI window creation:

```
display mdi-parent window
        background-low
        resizable
        lines          21
        size           70
        min-lines      21
        min-size       70
        title          "MDI Parent Window"
        system menu
        event          win-evt
        handle         h-mdi-parent

display mdi-child window
        upon           h-mdi-parent
        title          "MDI Child 1"
        line           3
        col            2
        size           30
        lines          10
        layout-manager lm-scale1
        resizable
        system menu
        handle         h-mdi-child-1.

display mdi-child window
        upon           h-mdi-parent
        title          "MDI Child 2"
        line           5
        col            37
        size           30
        lines          10
        layout-manager lm-scale1
        resizable
        system menu
        handle         h-mdi-child-2

display screen-1
        upon           h-mdi-child-1

display screen-2
        upon           h-mdi-child-2
```

Scrolling menu items

Selected menu items within menus and pop-up menus can now be scrolled. The scrolling behavior can be customized with new parameters implemented in the W$MENU library routine, as shown below:

Usage: CALL "**W$MENU**" using [WMENU-NEW|WMENU-NEW-POPUP]

[ScrollItems, FixedTopItems, FixedBottomItems, ScrollingInterval]

where:

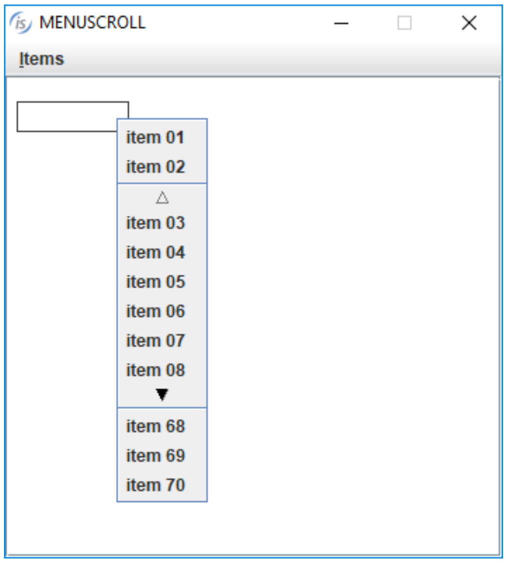
- ScrollItems is the number of visible items that can be scrolled
- FixedTopItems is the number of fixed and always visible items at the top of the menu
- FixedBottomItems is the number of fixed and always visible items at the bottom of the menu
- ScrollingInterval is the number of milliseconds used to automatically scroll menu items when hovering on the arrow icons

Code snippet:

```
CALL "W$MENU" USING WMENU-NEW-POPUP, 6, 2, 3, 300
              GIVING h-popmenu.
```

The results of the above code is shown on Figure 7, *Scrolling menu.*

Figure 7*.* Scrolling menu



C$DESKTOP routine

The new C$DESKTOP routine allows isCOBOL applications to launch associated applications registered on the native desktop to handle a file or URI.

Supported operations include:

- launching the user-default browser to show a specified URI;
- launching the user-default mail client with an optional mailto URI;
- launching a registered application to open, edit or print a specified file.

The new C$DESKTOP routine also supports "actions" that can be used to open, edit, e-mail or print a file.

Usage: CALL "**C$DESKTOP** " USING op-code, w-uri, [cs-flag]

Where:

op-code is numeric and valid values are defined in the isgui.def file: cdesktop-browse, cdesktop-edit, cdesktop-mail, cdesktop-open, cdesktop-print

URI is alphanumeric value, and is used as an argument for the op-code operation requested in the call.

CS-FLAG (numeric) is an optional parameter; when set to 1 the operation is executed on the client, otherwise it is executed on the server.

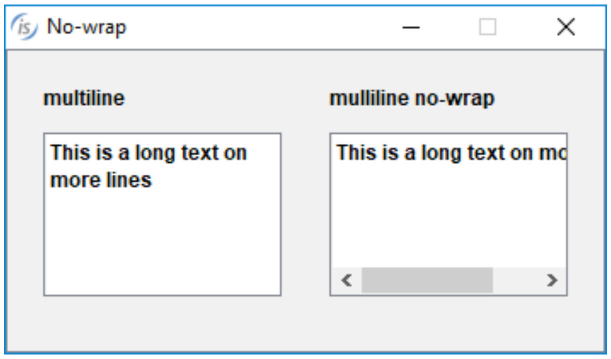
Code Snippet:

```
move "c:\tmp\myfile.png" to w-uri
call "C$DESKTOP" using cdesktop-open, w-uri
```

Other enhancements

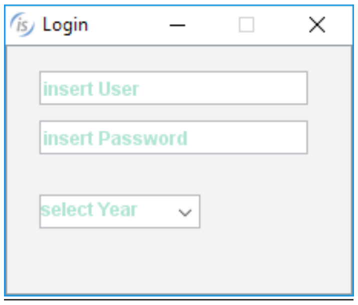
A new NO-WRAP style has been added to multiline entry-fields to avoid text wrapping on the next line, as shown on Figure 8*, NO-WRAP style*.

Figure 8*.* NO-WRAP style



The new configuration option iscobol.gui.placeholder_color is provided to set the color used for placeholder in entry fields, as Figure 9*, placeholder color* demonstrates.

Figure 9, placeholder color



The new MSG-GD-DBLCLICK event can be fired also on read only Grid component.

New methods have been implemented in the com.iscobol.rts.print.SpoolPrinter class to enhance customization of the Print Preview window.

- getSaveDefaultDirectory() to retrieve the Directory previously set
- setSaveDefaultDirectory(String path) to set the Directory shown in the Save dialog
- getSaveDefaultFilename() to retrieve the Filename previously set
- setSaveDefaultFilename(String path) to set the Filename shown in the Save dialog

Performance has been improved by an order of magnitude on sorted list-box and combo-box loading.

Framework improvements

isCOBOL Evolve 2017 R2 includes enhancements on XML and JSON stream handling, as well as several improvements to runtime configuration options and compatibility with other COBOLs.

Performance improvement on C$COPY library routine

C$COPY library routine has been optimized to improve performance by copying files, especially when runnning on an isCOBOL Server architecture to copy to and from clients.

See Figure 10, *C$COPY performance* for file transfer performance comparisons between isCOBOL 2017R2 and the previous version.

Figure 10, C$COPY performance

Type of CALL "C$COPY"	File size of file	isCOBOL 2017 R1	isCOBOL 2017 R2
copying a Binary file from server to server	about 90 MB	0.20	0.16
copying a LineSequential file from server to server	about 50 MB	0.20	0.16
copying an indexed file from server to server	about 40 MB	3.10	1.60
copying a Binary file from server to client	about 90 MB	8.50	3.90
copying a LineSequential file from server to client	about 50 MB	6.10	2.40
copying an indexed file from server to client	about 40 MB	126.00	38.50

New configuration options for XML and JSON streams

New configuration options are available for XML and JSON streams that can be used to configure the way XML and JSON streams are created in isCOBOL EIS.

- `iscobol.xmlstream.rtrim=true` – allows string trimming when generating XML streams. Defaults value is false.

- `iscobol.jsonstream.indent_number=n` – sets the number of white space characters to be used for JSON stream indentation. Default is -1

- `iscobol.jsonstream.omit_empty_elements=true` – allows empty JSON elements to be omitted from the stream. Default is false.

- `iscobol.jsonstream.rtrim=true` allows trimming of string in JSON streams. The default is false

Other configuration enhancements:

- The ***** wildcard is now supported in the configuration option `iscobol.code_prefix`, to allow loading all JAR files in a directory that contains COBOL programs, as shown below:

```
iscobol.code_prefix=/dir-classes:/dir-jar/*
```

New compatibility options

isCOBOL programs using character-based user interfaces can now take advantage of additional configuration options that provide better compatibility with other COBOL dialects, such as MicroFocus ®, RM/COBOL® and ACUCOBOL-GT®.

- `iscobol.terminal.data_range=minVal[,maxVal]` filters characters on the accept statement that are not in the specified ASCII range, converting them to spaces. For example, setting `iscobol.terminal.data_range=1` will automatically convert low-value (x"00") characters to space.
- `iscobol.terminal.no_autoclear=true` will prevent the `autoclear` function from executing on character accept statements without update
- `iscobol.terminal.numeric_autoclear=false` will prevent the `autoclear` function from running on numeric accept statements, to better support editing on numeric accepts with update, for example when editing dates with separator characters or numbers with decimal separator.

New routines have been implemented to provide higher compatibility with other COBOL dialects:

CBL_GET_SCR_SIZE, to return information about the size of the screen, **CBL_CLEAR_SCR**, to clear the whole screen using a specified character and attribute, **CBL_WRITE_SCR_N_CHAR**, to write a repeated character in a specific screen position, **CBL_WRITE_SCR_N_CHATTR**, to write a repeated character and attribute in a specific screen position.

The ISMIGRATE utility has been enhanced with a new option to strip the file extension from the destination filename. Migrated files with extensions stripped can be used in isCOBOL applications by setting the new Framework property `iscobol.file.index.strip_extension=true`

When this property is set to true, the file extension will be stripped from the physical file name declared in SELECT statement for indexed files, to match the file names generated in the migration process. Default value for this property is false.

isCOBOL Compiler and Debugger Enhancements

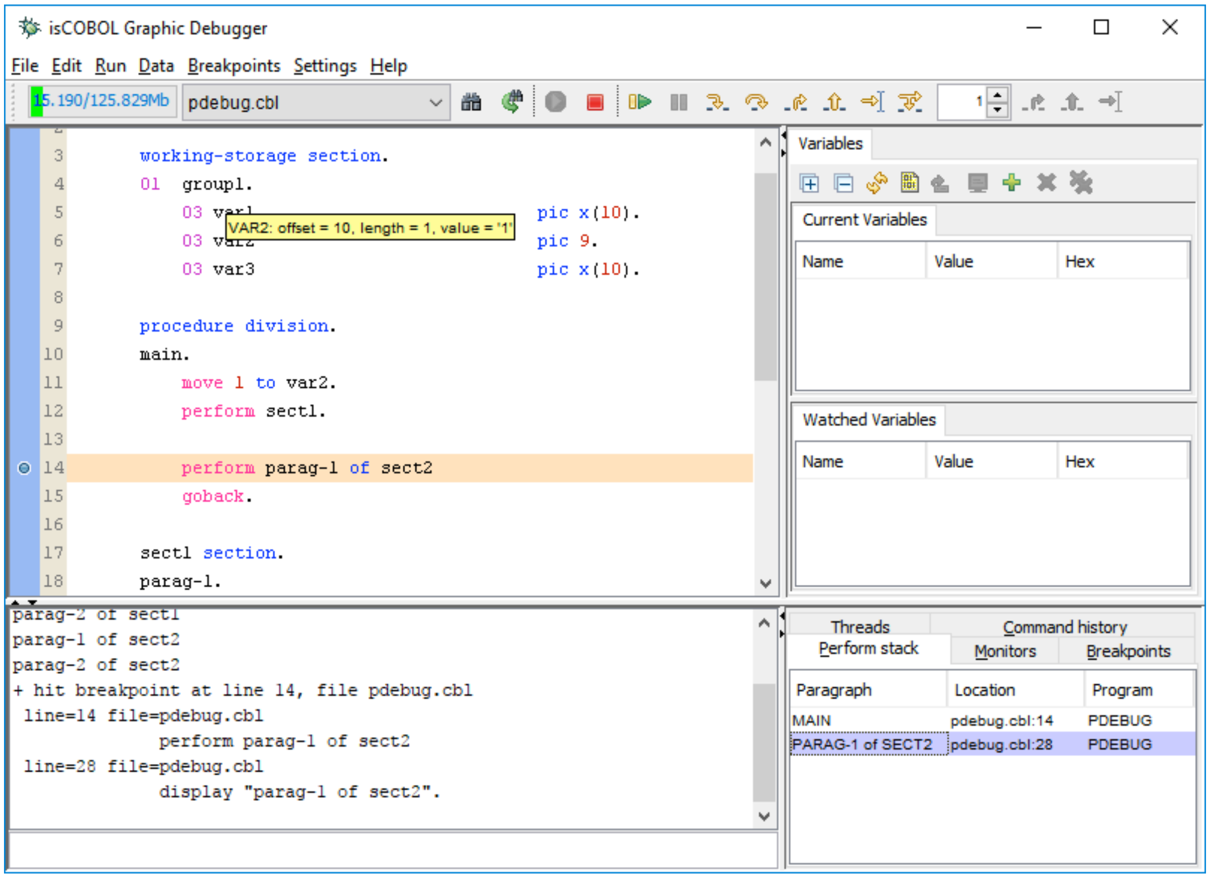
isCOBOL Evolve 2017 R2 includes changes to developer utilities and debugger to improve productivity.

Debugger enhancements

The isCOBOL debugger has been updated, improving variable hint examination, and providing an easy-to-read Perform stack view.

Figure 11, *Section names and variable hint in Debugger* shows the new display variable hint, visible when the mouse hovers on a variable, which now includes the offset, the length and value of the variable. Additionally, the Perform Stack pane now clearly shows the section name of the paragraph in the call stack, making it much easier to read when a COBOL program contains the same paragraph name in multiple sections.

Figure 11. Section names and variable hint in Debugger



```
 3        working-storage section.
 4        01  group1.
 5            03 var1                        pic x(10).
 6            03 var2                        pic 9.
 7            03 var3                        pic x(10).
 8
 9        procedure division.
10        main.
11            move 1 to var2.
12            perform sect1.
13
14            perform parag-1 of sect2
15            goback.
16
17        sect1 section.
18        parag-1.
```

VAR2: offset = 10, length = 1, value = '1'

```
parag-2 of sect1
parag-1 of sect2
parag-2 of sect2
+ hit breakpoint at line 14, file pdebug.cbl
 line=14 file=pdebug.cbl
             perform parag-1 of sect2
 line=28 file=pdebug.cbl
             display "parag-1 of sect2".
```

Name	Value	Hex

Name	Value	Hex

Paragraph	Location	Program
MAIN	pdebug.cbl:14	PDEBUG
PARAG-1 of SECT2	pdebug.cbl:28	PDEBUG

Stream2wrk utility

This new utility for COBOL developers replaces the previous xml2wrk and wsld2wrk utilities. It can now generate COBOL data structures to manage JSON files, and has been improved to handle XML with XSD schema files and WSDL definition.

The generated source code can also highlight fields declared optional in the WSDL or XSD schema files.

The utility can be run to generate data structure for JSON file as follows:

stream2wrk json uri [-o outputfile] [-p prefix] [-d]

To import a WSDL file:

stream2wrk wsdl uri [-o outputfile] [-v1.1]

To import an XML file:

stream2wrk xml uri [-o outputfile] [-p prefix] [-d]

Some usage samples are shown below:

```
stream2wrk json c:\dir\myfile1.json -o myjson1.def
```

```
stream2wrk wsdl http://ws.cdyne.com/ip2geo/ip2geo.asmx?WSDL -o
ip2geo.cpy
```

```
stream2wrk xml c:\dir\myfile2.xml -o myxml2.def
```

Database Bridge Improvements

isCOBOL Database Bridge generation has been rewritten in Java, and has been integrated in the isCOBOL compiler and available on all supported platforms. It's now a one-step process, and all needed files are generated at compile time. There is no need to run an external utility for each supported database anymore, as the compiler itself does it all automatically.

EDBI generation integrated in the compiler

New compiler properties have been added to configure the EDBI generation for supported DBMS (DB2, DB2_AS400, Informix, Oracle, MySQL, PostgreSQL, SQL Server:

```
iscobol.compiler.easydb=true|false (default: false)
iscobol.compiler.easydb.db2=true|false (default: false)
iscobol.compiler.easydb.db2_as400=true|false (default: false)
iscobol.compiler.easydb.informix=true|false (default: false)
iscobol.compiler.easydb.oracle=true|false  (default: false)
iscobol.compiler.easydb.mysql=true|false  (default: false)
iscobol.compiler.easydb.postgres=true|false  (default: false)
iscobol.compiler.easydb.sqlserver=true|false  (default: false)
iscobol.compiler.easydb.db2.prefix=...  (default: db2)
iscobol.compiler.easydb.db2_as400.prefix=... (default: d24)
iscobol.compiler.easydb.generic.prefix=...  (default: gen)
iscobol.compiler.easydb.informix.prefix=...  (default: ifx)
iscobol.compiler.easydb.oracle.prefix=...  (default: ora)
iscobol.compiler.easydb.mysql.prefix=...  (default: mys)
iscobol.compiler.easydb.postgres.prefix=... (default: pgs)
iscobol.compiler.easydb.sqlserver.prefix=...  (default: srv)
```

Further details on the configuration properties can be found on the isCOBOL 2017 R2 Documentation.

A new runtime property has been added to set the file name prefix of the generated EDBI programs:

```
iscobol.easydb.prefix=...
```

For example, to generate the EDBI programs for Oracle and MySQL, the compiler.properties file should contain

```
iscobol.compiler.easydb=true
iscobol.compiler.easydb.oracle=true
iscobol.compiler.easydb.mysql=true
```

When compiling the source code with the following command:

```
C:\Veryant\isCOBOL2017R2\sample\easydb>iscc -c=compiler.properties -
sp=..\isdef PROG-FILE1.cbl
Generated 'oraEDBI-file1.cbl'
Generated 'mysEDBI-file1.cbl'
```

Database Bridge classes are generated for both, Oracle and MySQL

At runtime, to execute the program and connect to a MySQL database, the runtime configuration should set the prefix as

```
iscobol.easydb.prefix=mys
```

while to connect to an Oracle database the following should set it as

```
iscobol.easydb.prefix=ora
```

Support for E type in EFD DATE

The EFD DATE directive now allows developer to store a COBOL 7-digit date in a database Date field. Such dates are represented in COBOL with the format YYYYEEE, where YYYY is the year, and EEE is the number of days since the first day of the year. As an example, the 1st of February of 2017 would be represented in COBOL as 2017032.

Database bridge can handle automatic conversion from YYYYEEE dates to the underlying database Date fields by specifying the following directives and field declaration:

```
$EFD NAME=ORDER_DATE
$EFD DATE=YYYYEEE
     03 customer-order-date pic 9(7).
```

The usual scenario for this configuration is when using code such as

```
accept customer-order-date from day YYYYDDD.
write customer-order-rec
…
read customer-order-file next…
```

Automatic conversion of the data from the database to the COBOL program and vice versa is carried out automatically by the Database Bridge engine.

IsCOBOL Server Improvements

isCOBOL Server has been enhanced to provide basic mirroring capabilities to allow transparent client redirection when a server is not available. Also the client was updated to allow multiple server definitions and to execute isCOBOL GUI utilities.

isCOBOL Thin Client connecting to more than one Application Server

The isCOBOL Thin Client has been enhanced to allow more IP addresses and ports to be specified in the command line, and the client will try to connect to the first available IP / port combination.

This can be useful, for example, when the Application Server host has several network connections available for redundancy, to ensure clients can connect to the server from the first available connection.

While this can be useful, especially when dealing with unreliable connections, it's not a replacement for isCOBOL Balancer, which enables true load balancing across multiple servers. True load balancing will allocate the most efficient server to a client connection, regardless of the order in which servers are specified in configuration, achieving better performance overall.

As an example, if an Application Server has two physical network connection with IP addresses ip1 and ip2, a Thin Client could be configured as follows:

```
iscclient -hostname ip1,ip2 -port 10999 MAINPROG
```

When ip1 is not available for connection, the client will automatically try to connect to ip2. If both ip1 and ip2 are not available, the client will return the message "Connection refused".

Several Application Servers can also be listening to different TCP ports, in which case the client can be configured to scan each one in turn, as shown below:

```
iscclient -hostname ip1 -port 10111,10112,10113 MAINPROG
```

The same feature can be enabled using the configuration settings on the client side by setting the properties:

```
iscobol.hostname=ip1
iscobol.port=10111,10112,10113
```

isCOBOL Thin Client utilities execution

isCOBOL Thin Client can now more easily execute all the framework utilities, after inserting the administrator password, without any extra configuration.

The supported utilities are:

- GIFE, Graphical Index, relative File Editor
- ISMIGRATE, Index file migration
- COBFILEIO, to generate object to access index file
- ISL, COBOL launcher
- CPK, Color Picker

For example, to run the `gife` or `ismigrate` utilities:

```
iscclient -hostname ip -port 10999 -utility gife
iscclient -hostname ip -port 10999 -utility ismigrate
```

isCOBOL EIS improvements

isCOBOL EIS has been enhanced with an improved REST Web Service Bridge generation, and WebDirect 2.0 has new features for grids and new routines.

REST web service

A single REST web service program can now manage both XML and JSON requests and responses, instead of two separate bridges.

EIS bridge programs can detect the correct incoming request format and the desired response format by automatically checking the Content-Type header in the HTTP request, and act accordingly. This means that a single web service program can now serve multiple clients regardless of data type format requirements, for example, a single web service can serve a front-end HTML / javascript application, which typically use JSON format, and a GUI desktop application that relies on XML format. Developers just need to write the program that implement the web service requests, and the Bridge will take care of the communication details.

This feature is enabled by default when generating programs with isCOBOL IDE 2017R2 when enabled the Service Bridge Editor, or compiling from command line when configured with the following properties:

```
iscobol.compiler.servicebridge=true
iscobol.compiler.servicebridge.type=REST
```

Developers not relying on Web Service Bridge generation, integrated in the compiler, can leverage the new methods `acceptEx(ICobolVar) and displayEx(ICobolVar)` of the HTTPHandler class that auto-detect the format reading the Content-Type of the HTTP request. Web Service now can read the HTTP request method used by the client (GET, POST, PUT, DELETE,…) with the new method `getMethod()`, which returns a string containing the requested method name. Request methods are usually used to indicate the desired action to be performed on the passed data.

If a request from a client does not specify a content type, developers can set the default format by setting the configuration

```
iscobol.rest.default_stream=xml|json (default json)
```

Additionally, the HTTPClient class has methods that read and write data using the format requested by looking at the Content-Type header (or the default if the Content-Type header is not used)

```
doPostEx(ICobolVar url, ICobolVar content)
getResponseEx(ICobolVar)
```

The client Beans generated by the compiler have been updated to use the new methods as well.

WebDirect 2.0 improvements

Grids in WebDirect 2.0 now have a pagination capability, in a web-style.

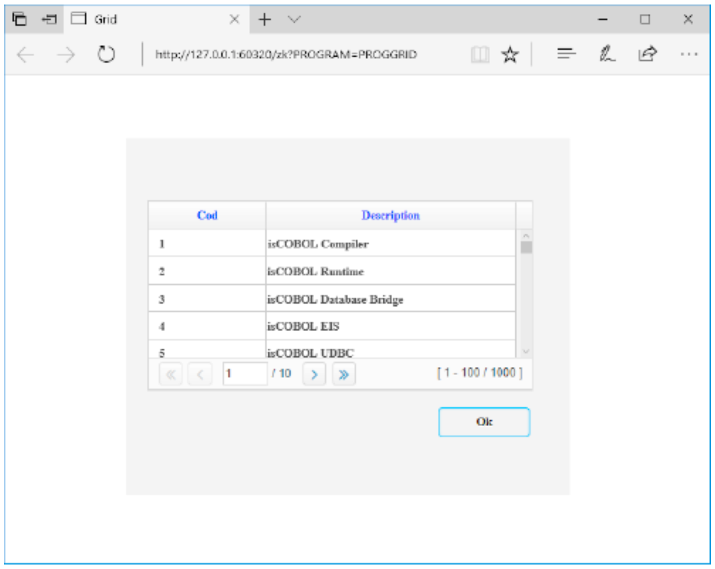
The new property ROWS-PER-PAGE has been added to the Grid component, to set the number of visible rows in the grid and to enable pagination. The COBOL program can load all the available data on the grid, and by setting ROWS-PER-PAGE to a number greater than zero, the runtime will activate automatic pagination. The browser will only display the visible rows, and the runtime will automatically request additional data as the user navigates the grid.

This new feature greatly improves performance and responsiveness of programs that use grids with a large number of records, and frees the developers from having to implement the navigation logic of a classic Paged Grid.

The following screen section snippet produces the grid shown in Figure 12, *Grids with ROWS-PER-PAGE in WD2.*

```
 03 gd1 Grid
    line 4 column 4 size 65 cells lines 7
    no-box centered-headings column-headings
    heading-color 10
    rows-per-page 100
```

Figure 12. Grid with ROWS-PER-PAGE in WD2



Cod	Description
1	isCOBOL Compiler
2	isCOBOL Runtime
3	isCOBOL Database Bridge
4	isCOBOL EIS
5	isCOBOL UDBC

Two new library routines: WD2$REDIRECT and WD2$EXECJS.

WD2$REDIRECT can be used to perform a redirect to a new URL.

Usage: CALL "**WD2$REDIRECT**" USING new-url, [target]

where target values are "_blank", "_parent", "_self", "_top" (default "_blank")

The typical use case of WD$REDIRECT is to allow links to be opened from program control, by redirecting the user on a landing page on program's termination, or to open a PDF document in a new browser page or tab, usually after a print job has been requested.

Code snippet:

```
move "http://www.veryant.com" to w-url
call "WD2$REDIRECT" using w-url "_self"

move "resources/pdf/customer-list.pdf" to w-url
call "WD2$REDIRECT" using w-url
```

WD2$EXECJS is used to run javascript code in a WD2 program.

Usage: CALL "**WD2$EXECJS**" using js-string

The routine accepts straight javascript code, with no script tags, and sends it to the browser for immediate execution in an optimized way.

Code snippet:

```
move "alert('hello world');" to js-string
call "WD2$EXECJS" using js-string
```

Additionally, WD2$EXECJS can be used to create new functions in javascript to send to the browser, to be later called by the COBOL program.

Code snippet:

```
move "function showError(message) alert(message);}" to js-string
call "WD2$EXECJS" using js-string
...
call "WD2$EXECJS" using “showError(‘Invalid customer code’);”
```