isCOBOL™ Evolve

isCOBOL Evolve 2018 Release 1 Overview

isCOBOL Evolve 2018 Release 1 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2018 R1.

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

isCOBOL EIS, umbrella of features to develop and deploy web applications, has been enhanced with isCOBOL WebClient, a new product that allows execution of isCOBOL Thin Client applications inside a web browser.

Starting from this version, Veryant provides a new scripting language named JOE (Java Object Executor) designed to ease migration of complex scripts that tie together COBOL programs. JOE is deeply integrated in the isCOBOL Framework and the Java ecosystem so isCOBOL IDE, now based on Eclipse Oxygen, provides a JOE plugin to write JOE scripts.

isCOBOL 2018 R1 includes several enhancements to the GUIs features , such as Notification windows and other new features and compatibility options.

Details on these enhancements and updates are included below.

isCOBOL EIS Improvements

isCOBOL EIS has been enhanced with a new product named Web Client. Several other minor enhancements designed to improve the existing capabilities of EIS.

WebClient

WebClient allows execution of isCOBOL Thin-Client applications, both graphical and characters based, in any web browser without modification. WebClient and Web Direct 2.0 can both create web applications, but they differ in how this is accomplished. Web Direct 2.0 allows the use of external CSS stylesheets and javascript files to spicy up the application, while WebClient is an easy way to render isCOBOL user interfaces in a web browser, retaining the look and feel of a desktop application. A WebClient server can host several isCOBOL applications, and users can be granted access to them individually.

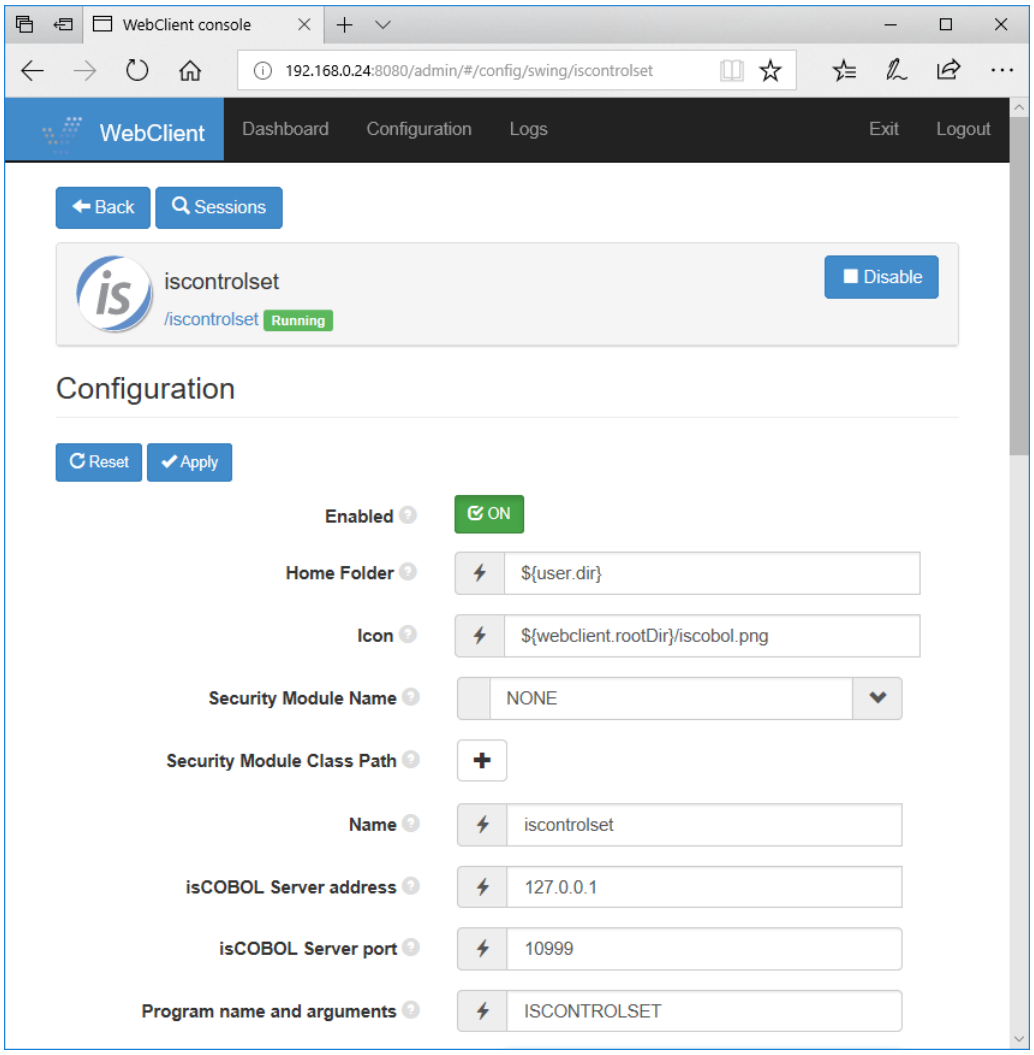
The isCOBOL program user interface is rendered on the browser, and user interaction, such as mouse moves, clicks and keyboard events, are sent back to the server for processing by the isCOBOL program. Communication between Thin-Client and the browser is handled using web sockets, and communication between the Thin Client application and the isCOBOL program is handled by the isCOBOL Application Server.

This new architecture brings some notable capabilities

- User can interact with the application as if it was a regular desktop application.
- Session resuming allows users to continue the same session after reconnecting or in case a lost connection is re-established.
- Administrators can monitor running applications in real time, viewing important information such as memory usage, CPU usage and response times.
- Administrators can provide assistance to end users by using the built-in remote assistance feature, that mirrors the user program on the WebClient administrative console, and allows administrator to take control of the session and help the user accomplish a task or troubleshoot a problem.

- Administration web console to configure users, isCOBOL programs and their settings, as shown in Figure 1, *Configuration page*.

Figure 1. Configuration page



The application running in the browser can be seen in Figure 2, *ISCONTROLSET sample running in the browser,* and Figure 3, *ISCONTROLSET sample running in the iPad's Safari browser.*

Figure 2. ISCONTROLSET sample running in the Microsoft Edge browser

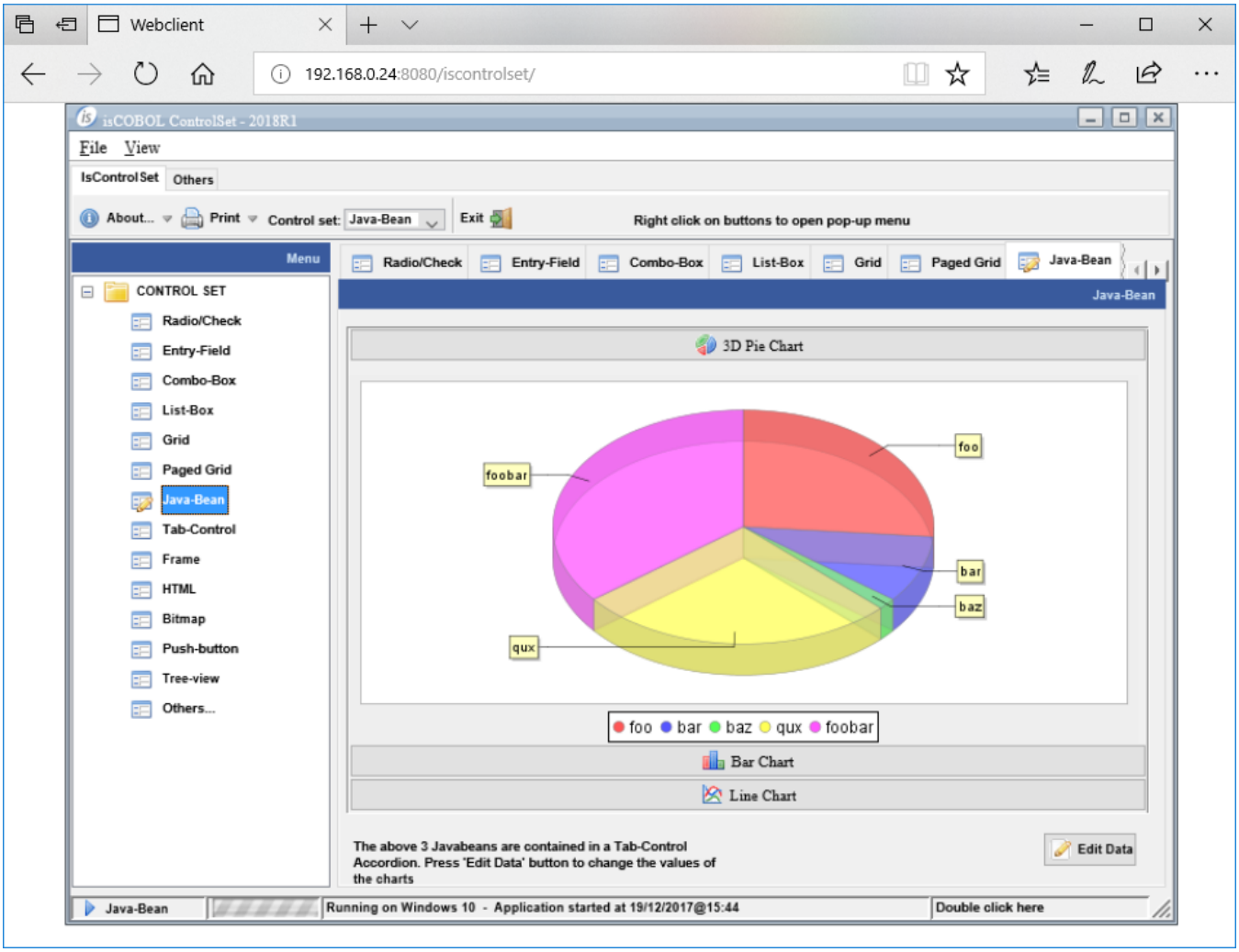


Figure 3. ISCONTROLSET sample running in the iPad's Safari browser

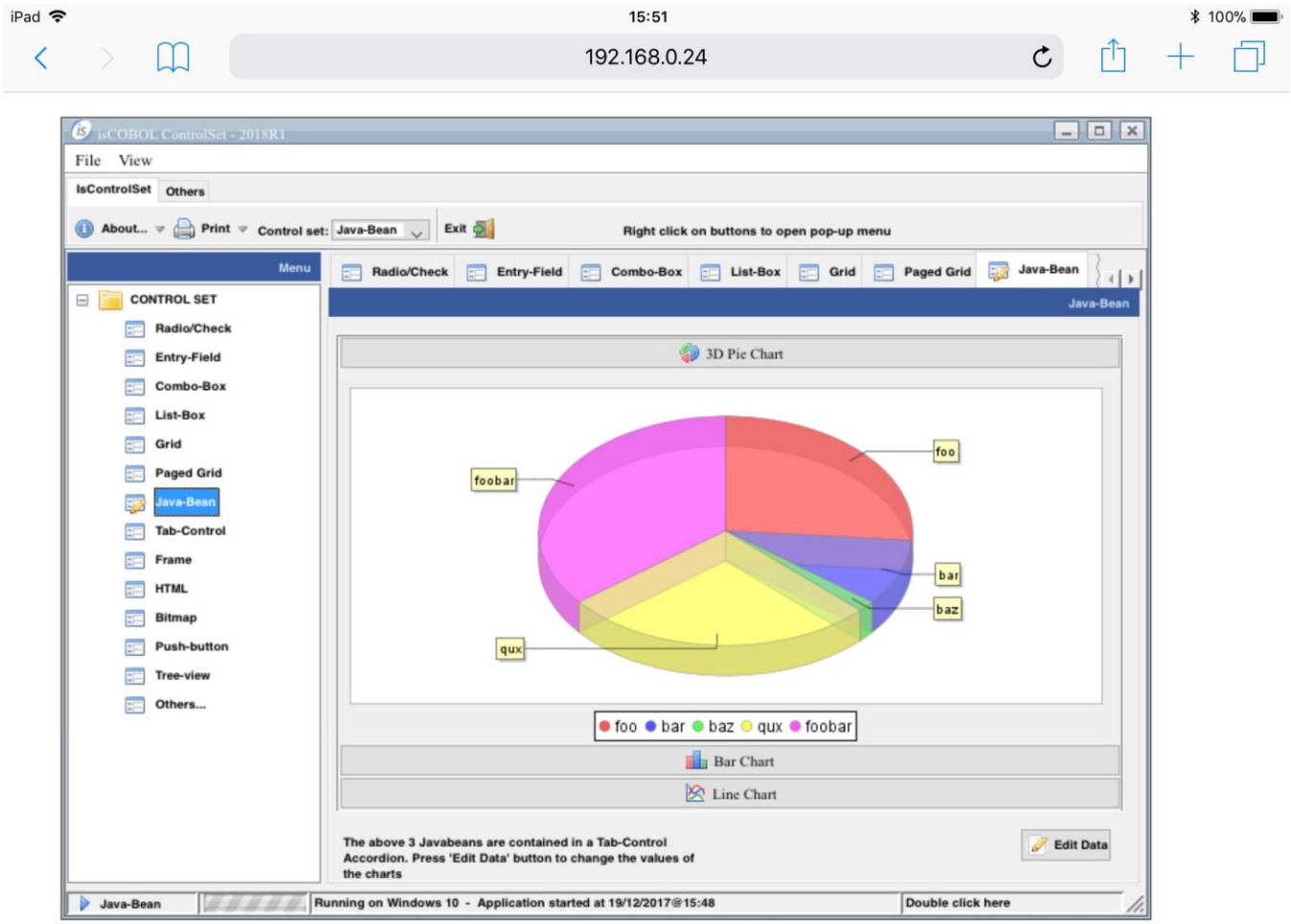


Figure 4, *Administrator view of running applications*, shows how administrators can monitor running processes on the WebClient server, and view or act on the user's screen, as shown in Figure 5, *Administrator view of user session*.

Figure 4. Administrator view of running applications

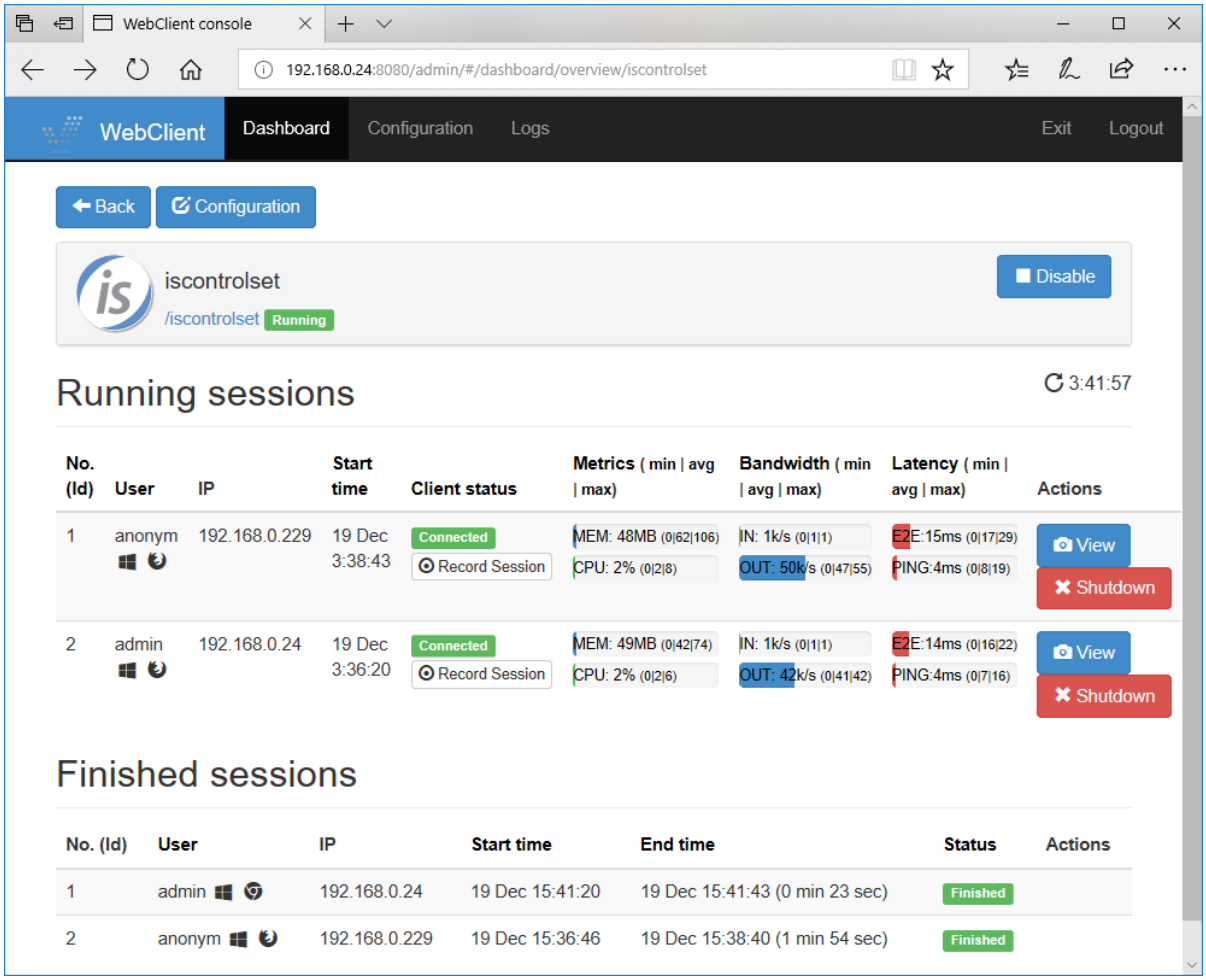
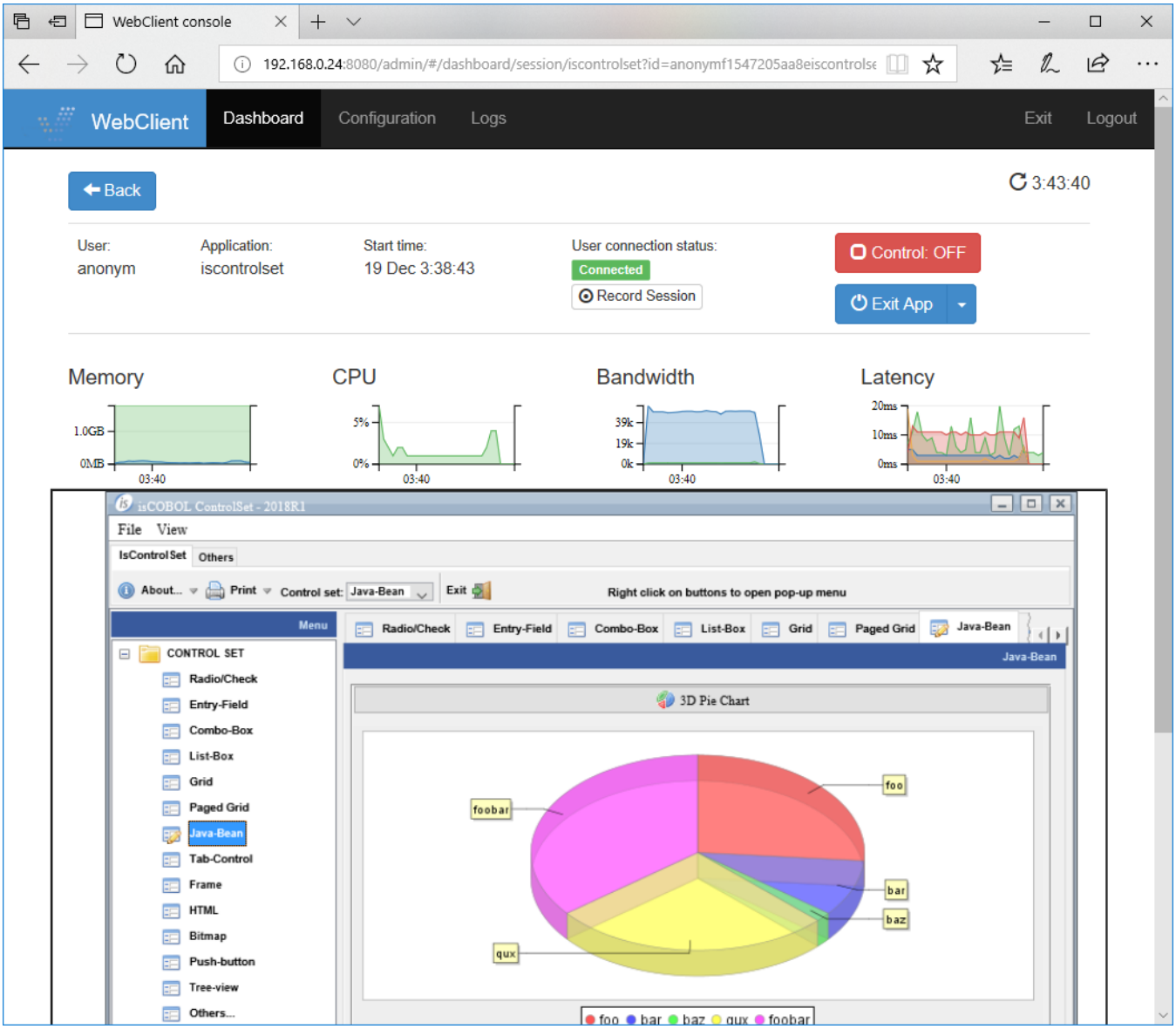


Figure 5. Administrator view of user session



Other Enhancements

The XML COBOL definition has been improved to support BASE64BINARY and HEXBINARY, allowing isCOBOL programs to consume web services that, for example, use MTOM communication.

The following is a code snippet that defines the field "a-document" to be handled as base64Binary:

```
01 soap-in-sendFileMtom identified by 'soapenv:Envelope'.
*> ...
   07 identified by 'nameAttachedFile'.
          08 a-nameAttachedFile pic x any length.
   07 identified by 'document' base64Binary.
          08 a-document pic x any length.
```

Certificate validation can now be turned off to simplify testing of web-services in non-production environments with a new configuration setting:

```
iscobol.http.ignore.certificates=true
```

When it is set to true the isCOBOL runtime will ignore certificates, if an SSL connection cannot be successfully established.

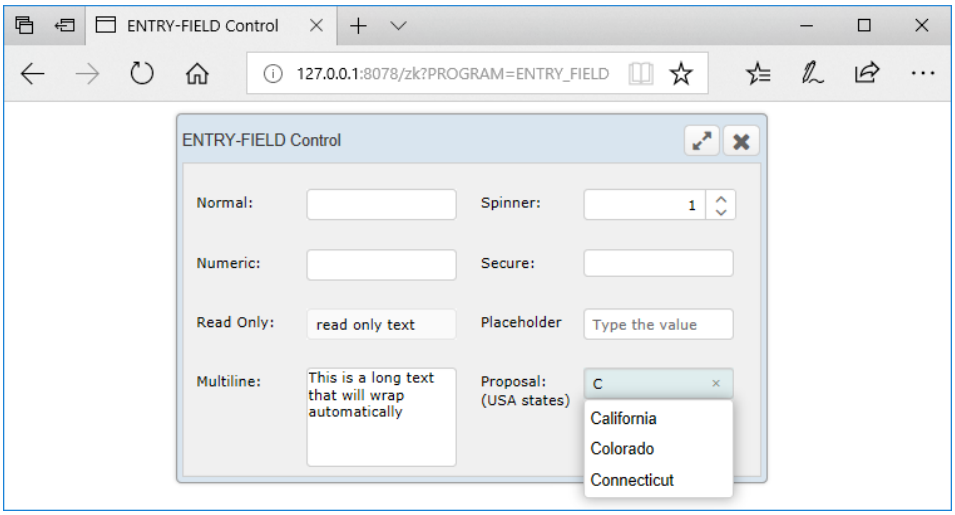
IsCOBOL Web Direct 2.0 now supports the PROPOSAL property on entry-field controls

Code snippet:

```
01 screen1.
    ...
    03 label line 11 col 38 lines 4 size 12 cells
       title "Proposal: (USA states)".
    03 entry-field line 11, col 51 size 19 cells
       proposal (..."California", "Colorado", "Connecticut", ...).
```

The results of the above code is shown on Figure 6, *Proposal in Web Direct 2.0*.

Figure 6. Proposal in Web Direct 2.0



JOE for isCOBOL

isCOBOL programs can be distributed from a server with isCOBOL Application Server; this is the architecture of choice for multiuser applications due to its many benefits, however, in order to get the best results, it requires all the processing made using isCOBOL environment.

Many legacy applications are developed intermixing COBOL programs and interpreted scripts of some kind (e.g. Bourne shell); in order to get these applications running in isCOBOL Application Server, currently, it is necessary to translate the scripts into COBOL programs and compile them.

This process can be lengthy and eventually the results could be less appealing than the starting point because:

- interpreted procedures are now compiled procedures
- procedures written with a language oriented to manage operating system tasks are now written using COBOL (a Business Oriented Language).

So, in order to speed up the migration process getting at the same time a better result, it would be useful to have a scripting language whose features are:

- ability to access any isCOBOL/Java resource: isCOBOL (as well as Java) is operating system independent therefore the Java environment is its virtual operating system
- easy to change in order resemble any scripting language in terms of capability and readability
- easy to customize in order to get frequently used operations at hand
- easy to extend in order to be useful for future applications' enhancements, not only for the migration process;
- easy to understand and use;
- 100% compatible with the isCOBOL Application Server architecture.

As an example, the following JOE script asks the user for input, and then displays a salutation:

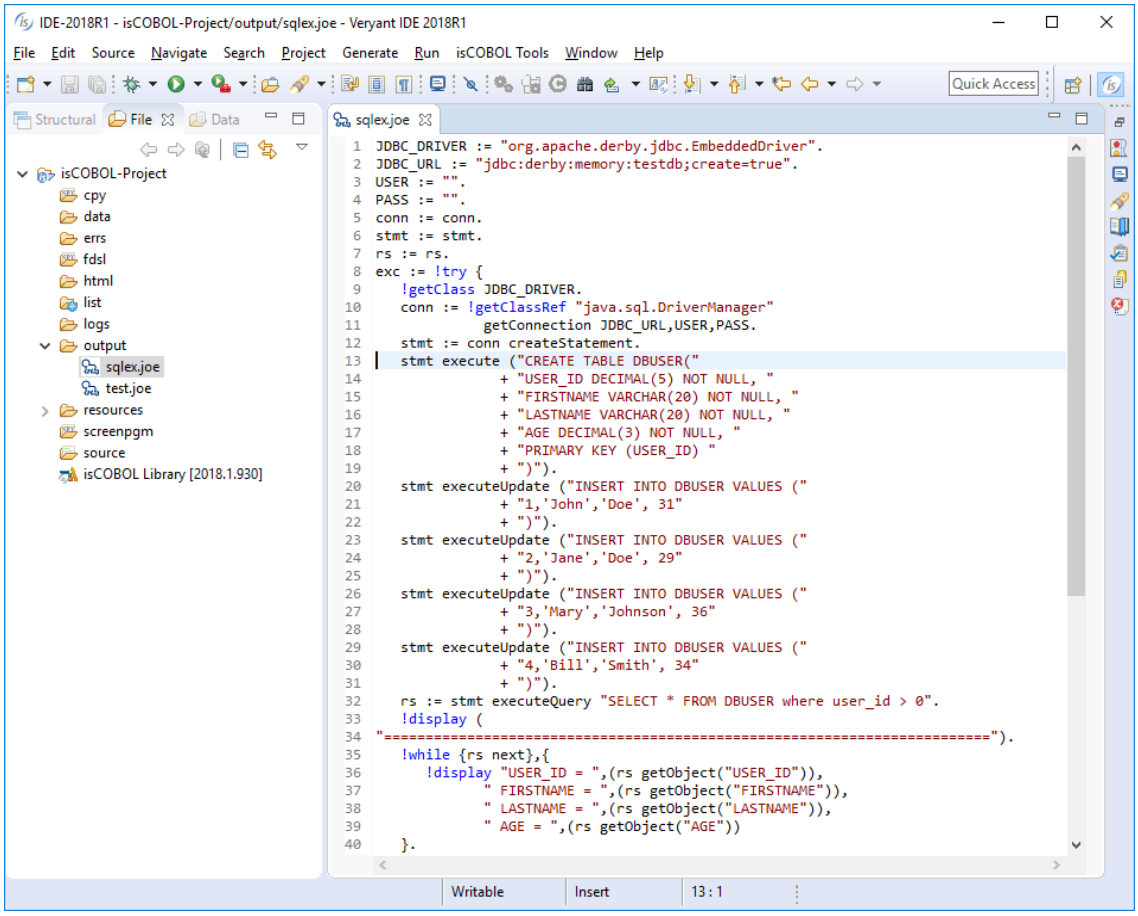
```
/** JOE says Hello */
!display "What's your name?".
VAR := !accept.
!display "Hello ",VAR.
!exit.
```

and it can be executed by running:

```
iscrun -joe test.joe
```

In Figure 7, *JOE JDBC Example*, shows a more complex script that creates a JDBC connection to a JDBC database and runs basic SQL statements. This example can be executed without any compilation. A new isCOBOL IDE plugin is provided to simplify JOE scripts coding.

Figure 7. *JOE JDBC Example*



isCOBOL IDE Enhancements

The isCOBOL 2018 R1 IDE is now based on the new Eclipse Oxygen release, which includes many improvements in functionality and performance.

isCOBOL Screen Painter now allows the definition of chaining parameters and customized linkage that affects the generated source code.

Eclipse Oxygen

Eclipse Oxygen is based on Java 8 and offers many advantages compared to the previous versions:

Support for high DPI monitor, such as Apple's retina screens and 4K displays, improving on the usability of toolbar icons.

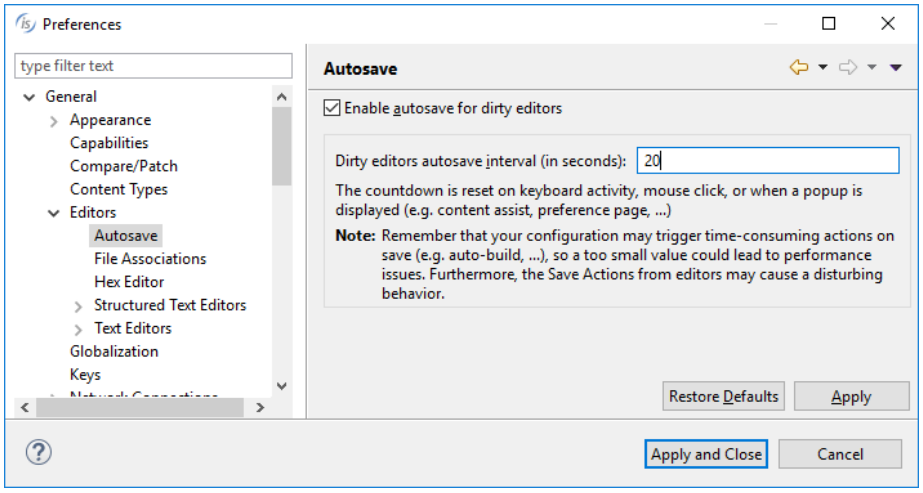
GTK3 support for Linux platforms.

Autosave to automatically save all open files in the isCOBOL editors. The autosave countdown is reset on any user activities, such as keystrokes and mouse event. See Figure 8, *Autosave feature.*

Smart projects import and automatic missing editor installation.

The new Launch Group configuration type allows you to launch multiple configurations sequentially, with configurable actions after launching each group member.

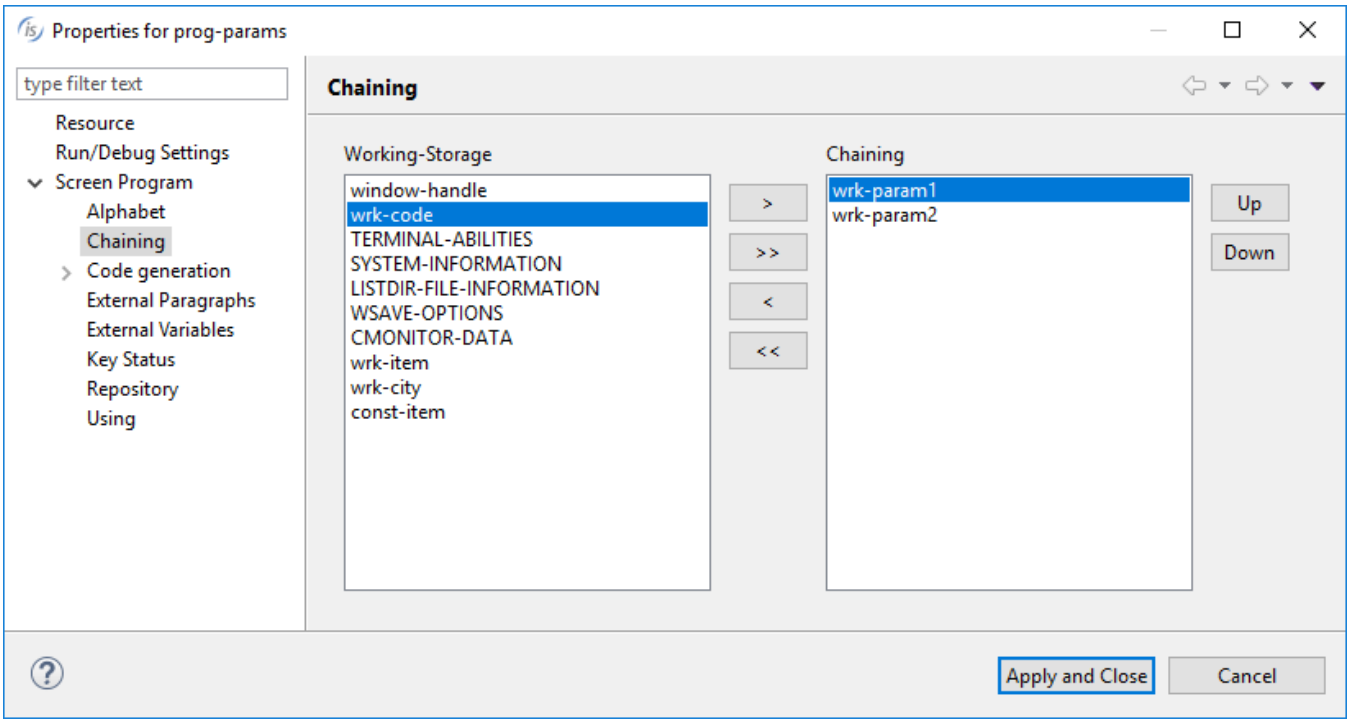
Figure 8. Autosave feature



Chaining parameters

The Program Properties of an isCOBOL Screen Program now allows defining a list of parameters that the program needs to receive from the command-line, as shown on Figure 9, *Chaining parameters*. Moreover, it's also possible to define a customized list of linkage parameters. All data items declared in linkage section are generated in the "procedure division using" by default, but each linkage data item can be individually selected for inclusion in the USING clause.

Figure 9. Chaining parameters



The following is the generated code:

```
procedure division chaining wrk-param1 wrk-param2.
```

New User Interface Features

Notification windows have been implemented to advise the end user of events with an elegant and convenient pop up window. Gradient effects can be applied on existing window and multiple monitors are now fully and easily supported.

Several other minor enhancements designed to improve User Interface handling.

NOTIFICATION windows

Notification windows are useful to provide users with feedback on the outcome of programs execution, such as informing a user that a computation has completed, that a print job has been carried out, or that a form contains errors.

A notification window uses a customizable corner of the desktop to display notifications or status information.

Notification windows can be used to show information that does not require user interaction, showing only a simple message or a specific screen section. If user interaction is required, there is no need to accept the notification screen, and the accept on the current window will terminate with the exception-value generated by the controls on the notification window. The notification window can also automatically close after a timeout, specified in the BEFORE TIME property.

The following code snippet will show the notification window shown in Figure 10, *Notification window in Microsoft Windows*, and Figure 11, *Notification window in Mac OSX*.

```
display notification window
        bottom right
        before time 500
        lines 5 size 40
        handle h-notification
display mask-notification upon h-notification
```

Figure 10. Notification window in Microsoft Windows

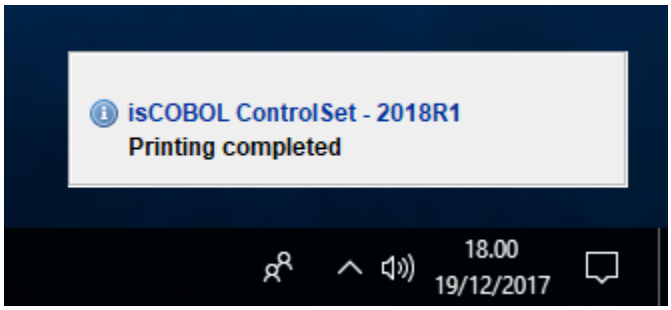
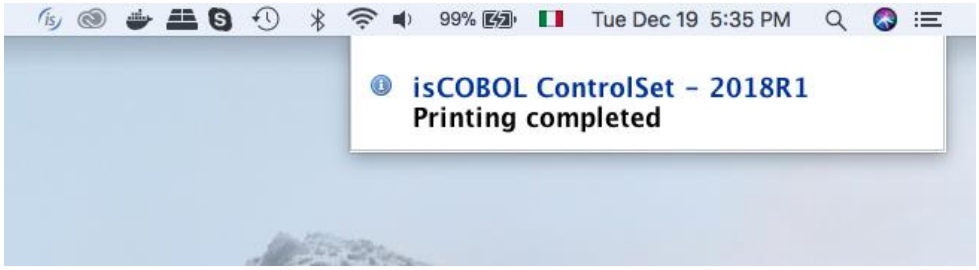


Figure 11. Notification window in Mac OSX



Gradient effects

Window controls can now display gradients as background color using the two new properties

GRADIENT-COLOR-1 specifies the starting color, and supports RGB notation

GRADIENT-COLOR-2 specifies the ending color , and supports RGB notation

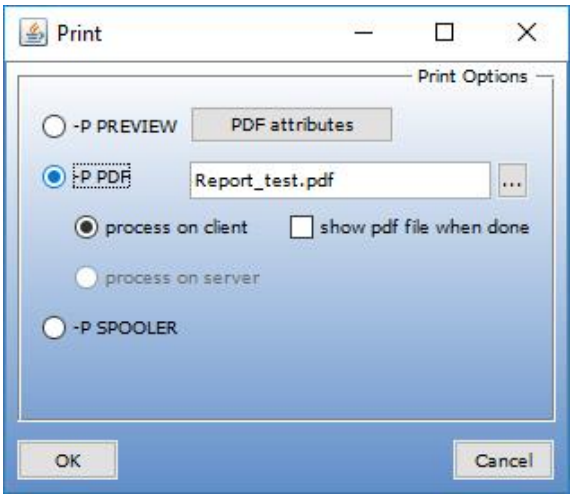
GRADIENT-ORIENTATION specifies the orientation of the gradient effect. Allowed values are: gradient-north-to-south, gradient-northeast-to-southwest, gradient-east-to-west, gradient-southeast-to-northwest, gradient-south-to-north, gradient-southwest-to-northeast, gradient-west-to-east, gradient-northwest-to-southeast.

Code snippet to create a window with gradient effects:

```
display independent graphical window
     title "Print"
     lines 15 size 52
     handle h-win-print
     gradient-color-1 rgb x#ffffff
     gradient-color-2 rgb x#6D8AD6
     gradient-orientation gradient-north-to-south.
```

The result of the above code is shown on Figure 12, *Gradient on window*.

Figure 12. Gradient on window



Multiple monitors

The new C$MONITOR routine allows isCOBOL applications to query the system about the number of attached monitors, get the screen resolution of each, their relative positioning, and to find which one is the primary display.

Usage: CALL "**C$MONITOR**" USING op-code, other-parameters

Where:

op-code is numeric, and valid values are defined in the iscobol.def file: cmonitor-get-no-monitor and cmonitor-get-monitor-info

other-parameters depends on the op-code used.

The op-code cmonitor-get-no-monitor returns the number of monitors available, and the number of the primary monitor.

The op-code cmonitor-get-monitor-info returns configuration information of a specific monitor.

The returned data is shown below:

```
01 cmonitor-data.
   03 cmonitor-usable-screen-height          pic x(2) comp-x.
   03 cmonitor-usable-screen-width           pic x(2) comp-x.
   03 cmonitor-physical-screen-height        pic x(2) comp-x.
   03 cmonitor-physical-screen-width         pic x(2) comp-x.
   03 cmonitor-start-y                       signed-int.
   03 cmonitor-start-x                       signed-int.
```

The DISPLAY WINDOW statement has a new SCREEN-INDEX property that can be used to set the monitor that will be used to display the window.

The W$CENTER_WINDOW library routine has been enhanced to allow specification of a monitor with an additional parameter.

The following code snippet shows how to inquire the number of monitors, read the configuration information of the second monitor, and create an independent window centered on the second monitor:

```
call "C$MONITOR" using cmonitor-get-no-monitor
                       number-of-monitor
                       primary-monitor
call "C$MONITOR" using cmonitor-get-monitor-info
                       2
                       cmonitor-data
display independent window lines 30 size 100
        title "Window on second monitor"
        screen-index 2
        handle in win2

call "W$CENTER_WINDOW" using win2, 2
```

Other enhancements

New configuration properties have been implemented to customize editing on GUI controls:

iscobol.gui.kbd_case=lower/upper used to force characters casing on GUI controls

iscobol.gui.entryfield.implied_decimal=true to have implied decimal values on entry-fields, based on the number of decimal digits defined in the picture in the associated variable of the entry field.

The TRANSPARENT style, previously available only on label controls, is now supported on check-box, radio-button and frame. This allows the creation of user interfaces with gradients as background colors.

Code snippet:

```
01 screen-1.
   03 bitmap line 1 column 1 size 100 cells lines 20 bitmap-handle bmp-handle.
   03 frame transparent
      line 2 column 3 size 56 cells lines 10 cells title "Frame".
   03 label transparent
      line 4 column 6 size 10 cells title "label".
   03 check-box transparent
      line 6 column 6 size 20 cells title "check-box".
   03 radio-button transparent   group 1 group-value 1
      line 8 column 6 size 20 cells title "radio-button1".
   03 radio-button transparent   group 1 group-value 2
      line 8 column 30 size 20 cells title "radio-button2".
```

The result of the above code is shown on Figure 13, *Transparent controls*.

Figure 13. Transparent controls

Framework improvements

New library routines and configuration settings

isCOBOL supports the following new library routines:

C$GETRUNENV, to get information about the runtime environment the isCOBOL application is running on.

Usage: CALL "**C$GETRUNENV**" GIVING w-env

Where w-env is a numeric variable and it will be set according to the environment where isCOBOL is running, possible values are declared in iscobol.def:

```
78 runenv-standalone                    value 1.
78 runenv-charva                        value 2.
78 runenv-remote-call                   value 3.
78 runenv-thin-client                   value 4.
78 runenv-web-client                    value 5.
78 runenv-wd2                           value 6.
78 runenv-j2ee                          value 7.
78 runenv-mobile                        value 8.
```

C$UNLOAD_NATIVE, to unload a previously loaded native library

Usage: CALL "**C$UNLOAD_NATIVE**" USING "library_name".

New configuration settings allow the remapping of the sqlcode returned by ESQL, and to support the DCI Connector to be used in heavy multi-threaded environments

- `iscobol.esql.sqlcode.{value}=newValue` remaps the sqlcode returned by ESQL access. This enhances the compatibility with other ESQL running with different COBOL pre-compilers. For example, setting

  ```
  iscobol.esql.sqlcode.100=1403
  iscobol.esql.sqlcode.1843=-1843
  ```

  allows to remap the sqlcode 100 and 1843 to 1403 and -1843 respectively. This simplifies the migration from Pro*COBOL with Oracle database that need those specific codes.

- `iscobol.file.index=dcic` allows access to DBMaker tables through the DCI Connector

- `iscobol.file.connector.program.dcic=/path/dcic` specify where the dcic executable file is installed.

The isCOBOL log has been enhanced by adding the ability to include date and time information in both the log file path and the file name. This is useful when logging on isCOBOL Application Server or a J2EE server, where running applications usually generate several log files. For example, by configuring:

```
iscobol.logfile=/tmp/%yyyy%mm/%dd/%hh/iscobol-%hh.%nn.%ss.log
```

the runtime will generate iscobol-15.35.30.log file in the /tmp/201712/21/15 folder. This simplifies searching the log files, since they are grouped inside folders for each single day and hour.

The Logger object is now accessible to isCOBOL programs, allowing them to write additional information with customizable log levels. For example, the following code snippet:

```
repository.
    class IsRuntime as "com.iscobol.logger.LoggerFactory"
    class MyLogger  as "com.iscobol.logger.Logger"
    .
working-storage section.
77 mylog object reference MyLogger.
procedure division.
main-logic.
    set mylog to IsRuntime:>getCurrLog()
    if mylog not = null
       mylog:>warning("this is a warning")
       mylog:>info("this is a info")
       mylog:>severe("this is a severe")
    end-if
```

will log the following lines:

```
21-dic-2017 16.48.34.112 INFO: ENTER PROGRAM 'PROGLOG' {
21-dic-2017 16.48.34.245 WARNING: this is a warning
21-dic-2017 16.48.34.245 INFO: this is a info
21-dic-2017 16.48.34.245 SEVERE: this is a severe
21-dic-2017 16.48.34.245 INFO: EXIT PROGRAM 'PROGLOG' }
```

New compatibility options

isCOBOL now provides even better compatibility with other COBOL dialects, such as MicroFocus ®, RM/COBOL® and ACUCOBOL-GT®.

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

CBL_CREATE_FILE, to create a new file and leave it open for operations

CBL_OPEN_FILE, to open an existing file for processing

CBL_CLOSE_FILE, to close an open file

CBL_READ_FILE, to read bytes from a file

CBL_WRITE_FILE, to write bytes to a file

CBL_FLUSH_FILE, to ensure all buffers for a file are written to disk

CBL_TOLOWER, to convert a data item to lower case

CBL_TOUPPER, to convert a data item to upper case

C$CARG, to read parameter information on the received parameters given its name

C$DARG, to read parameter information on the received parameter given its position

C$CENTURY, to retrieve the first two digits of the current year

C$DELAY, to suspend the running program without using CPU resources

DELETE, to delete a file

A new op-code **winprint-set-job** is now supported in WIN$PRINTER routine to manage concurrent printer jobs in multi threaded programs.

A new compiler option -sv has been implemented to support variable source format.

isCOBOL Compiler Enhancements

isCOBOL Evolve 2018 R1 implements new compiler options and new syntax to improve productivity.

New compiler options

-wmwc to show warnings for long variables in MOVE... WITH CONVERT statements.

This option is useful for developers that need to check at compile time the MOVE statements that may produce wrong runtime behavior when a type conversion may cause data truncation.

-watn to show warnings when moving alphanumeric variables to numeric variables. This option is useful to check at compile time MOVE statements used to assign alphanumeric variables to numeric ones, which are unpredictable without the –cudc compiler option.

New syntax supported

BINARY(n) syntax can now be used on group level and is applied on all child elements.

As an example, the following variable declaration:

```
01  ARG-DESCRIPTION BINARY(2).
       02 ARG-TYPE        PIC 99.
       02 ARG-DIGIT-COUNT PIC 99.
       02 ARG-SCALE       PIC S99.
```

is equivalent to:

```
01  ARG-DESCRIPTION.
       02 ARG-TYPE        PIC 99  BINARY(2).
       02 ARG-DIGIT-COUNT PIC 99  BINARY(2).
       02 ARG-SCALE       PIC S99 BINARY(2).
```

TYPEDEF syntax is now supported. It allows to define a type to be used in other variable declaration. It is similar to the existing SAME AS clause, but the TYPEDEF variable does not actually define a variable, but only a type definition for other variables to use.

Code snippet to define a type definition and use it to define 2 variables:

```
77  type1      PIC  9(9)v9(6) IS TYPEDEF.
01  var1       usage type1.
01  var2       usage type1.
```

isCOBOL Server Improvements

isCOBOL File Server has been enhanced with the new ISF protocol to be used in file name assignments. This allows file access on multiple isCOBOL File Servers running on different hostnames.

The syntax to be used for ISF protocol is:

```
isf://hostname[:port]:path/to/file
```

Where:

- hostname is the server name or IP address where the File Server is listening
- port is the port where the File Server is listening. If omitted, the default port 10997 is used
- path/to/file is the name of the remote file to open.

This syntax is supported in multiple scenarios:

- SELECT on a specific file. As an example, the following select declares FILE1 handled by the File Server listening on 192.168.0.1 on the default port:

```
select file1 assign to "isf://192.168.0.1:/usr/data/file1"
       organization indexed
       access dynamic
       record key file1-key.
```

- Configuration can be applied on multiple files depending on configuration rules, for example to affect all files that have a relative path:

```
iscobol.file.prefix=c:/tmp;isf://192.168.0.2:10123:/usr/data
```

  The isCOBOL framework will test the presence of the file locally first, in c:\tmp folder. If the file is not found, a second attempt is made using the File Server listening on 192.168.0.2 on port number 10123.

- Library routines that manage files: C$COPY, C$DELETE and C$FULLNAME. For example, this code snippet shows how to copy a file from the local Windows machine where isCOBOL is running in standalone to the remote Linux machine where the isCOBOL File Server is running:

```
call "C$COPY" using "c:\tmp\file-orders"
                    "isf://192.168.0.1:/usr/data/file-orders"
```